

GM-Search

A System for Stereochemical Substructure Search

Inaugural-Dissertation

Submitted for the
degree of Doctor of Philosophy
to the Philosophical Faculty II
of the
University of Zürich

by
Bernhard Rohde
from the Federal Republic of Germany

Approved by Prof. Dr. A.S. Dreiding and Prof. Dr. V. Strassen

Zürich 1988
Zentralstelle der Studentenschaft



GM-Search

A System for Stereochemical Substructure Search

Inaugural-Dissertation

Submitted for the
degree of Doctor of Philosophy
to the Philosophical Faculty II
of the
University of Zürich

by
Bernhard Rohde
from the Federal Republic of Germany

Approved by Prof. Dr. A.S. Dreiding and Prof. Dr. V. Strassen

Zürich 1988
Zentralstelle der Studentenschaft

Acknowledgement

Many individuals have influenced the work described in this thesis or contributed to its appearance. I wish to thank Prof. Dr. A. S. Dreiding for acting as the condensation nucleus for his research group on the fascinating field of non-numeric computer applications in chemistry and for the freedom to develop my own ideas. My co-workers in this group, namely Dr. R. Custer, Dr. P. Floersheim, Prof. Dr. H. R. Haegi, J. Inglis, H. Jost, Dr. M. Marsili, and Dr. F. Seaman, always had an open ear for the discussion of even the most foolish ideas.

Special thanks are due to Roland 'Nonius' Custer, from whom and with whom I learned a lot, be it mathematics, computer science, swiss history and politics, or any other topic. The main parts of the DRAWEDITOR for the input of molecular structures were written by him, and I also gladly remember the many hours of jointly solving computer puzzles called debugging. I am also specially indebted to Prof. Dr. H. R. Haegi for his help in clarifying my definitions and for many helpful suggestions.

Dr. M. Huber deserves my gratitude for posing the challenging problem of building a structure retrieval system based upon the relational system.

Besides these former and present members of the computer group of Prof. Dreiding, I am very much obliged to Prof. V. Strassen and Dr. M. Fürer for embarking on this interdisciplinary subject from the mathematical bank of the river.

The typographical appearance of this dissertation was made possible by use of the SCRIPT text processing facilities available at the computing center of the University of Zürich and by the collection of SCRIPT macros written by Dr. M. Karpf. The statistical computations and the corresponding plots were performed using SAS and SAS/GRAPH. The drawings were done by

MacDraft on a Macintosh personal computer. Without these resources the frequent revisions of the text would have been impossible.

Finally, I want to thank all the many unmentioned members of the Organic Chemical Institute for their co-operative attitude, which made it very pleasant to work at the University of Zürich.

This work was supported by the Swiss National Science Foundation and by the Ciba-Geigy AG/Basel.

To my parents

To Monika



Digitized by the Internet Archive
in 2026

TABLE OF CONTENTS

1. Introduction	1
1.1. The Problem	1
1.2. Organization	2
1.3. Current Structure Retrieval Systems	3
2. The Users View of the System	5
2.1. User Interface	5
2.2. Graphical Input	6
2.3. Contents of the Database	7
3. The Relational System	9
3.1. M-Descriptions	9
3.2. Generalized M-Descriptions	13
4. Algorithms	23
4.1. Structure Pruning	23
4.2. Preprocessing	28
4.2.1. Computing Ring Size Information	29
4.3. Canonizing	35
4.3.1. Refinement	37
4.3.2. Refinement of Graphs	37
4.3.3. Refinement of GM-Descriptions	41
4.3.4. The Automorphism Group of Symmetrical Molecules	48
4.3.5. Straightforward Canonization Procedure	52
4.3.6. Canonization Using Generators	57
4.4. Substructure Testing	65
4.4.1. Subgraph Isomorphism Testing	65
4.4.2. Relaxation for Sub-GM-Testing	72
4.4.2.1. Initialization	74
4.4.2.2. Refinement of Mappings	77
4.4.2.3. Case Analysis	82
4.4.2.4. Improvements	86

4.5. Screening	86
4.5.1. Superimposed Coding	87
4.5.2. Screen Types Used in GM-Search	90
4.5.3. Element-Count Screens	91
4.5.4. Type-of-Ring Screens	92
4.5.5. Atom-Bond-Sequence screens	95
4.5.6. Matching Screen Generation	98
5. Implementation	102
5.1. The File Structure	102
5.2. The Library Adder	104
5.3. The Library Searcher	105
5.4. The Screen Managers	106
6. Performance	110
6.1. Comparison of PERQ-2, IBM-PC/AT-2, and DEC-10	111
6.2. Pruning	114
6.3. Ring Perception	116
6.4. Canonizing	117
6.5. Screen Generation	119
6.6. Searching	120
7. Areas of Further Development	126
7.1. Improvements of GM-Search	126
7.2. Scope of the Data Structure and the Key Algorithms	127
7.3. Extensions to the Data Structure	129
8. Summary of Major Results	130
Appendix I: Glossary	132
Appendix II: Sample Search	136
References	141

"You must know, Cleverbyte, Megachip has had the greatest idea ever: By making chips work faster than light you can read out your computed result virtually before you insert your data, provided you position your output station at a location remote from your input device."

'Professor Cleverbytes visit to heaven'
by Niklaus Wirth

1. Introduction

1.1. The Problem

Today, searching through literature is a major part of chemical research. For this task certain data bases have been devised. They can be searched either by keyword or by chemical structure. Most valuable is the possibility to find references, which deal with molecules containing an interesting substructure. Some already existing data bases provide this facility. They mostly handle what is called the constitution or chemical graph [1] of a molecule.

In accord with the modern development of chemistry, however, it is desirable to distinguish between different stereoisomers. In our research group [2], this problem is tackled with a concept called 'relational system' and with the 'M-descriptions' based upon it. In continuation of this approach, this thesis is concerned with the problem of storing stereochemically described chemical structures and retrieving them by substructure search. The M-descriptions have been generalized to 'GM-descriptions' to comprise also substructures. The resulting structure retrieval system is called 'GM-Search'.



It was intended to devise a method which, on the constitutional level, is of similar efficiency as those used in existing database systems and which is able to handle the stereochemical aspect with only a small detriment in performance. GM-Search is tested with the conventional types of stereochemical information, such as tetrahedral centers and double bonds, but according to the theory of M-descriptions, it can be extended to describe also other stereochemical relationships.

1.2. Organization

Building a structure retrieval system is a complex task and many problems have to be solved.

- A user interface program must be devised for convenient input of molecules and query substructures.
- A data structure must be defined to describe and store structures and substructures.
- Algorithms must be developed to allow efficient updating and searching of the library of structures.
- A set of programs and files must be constructed to implement the actual structure retrieval system.

Because of the variety of these topics, each is treated in a separate chapter; the relevant literature will be cited there.

An additional chapter discusses the computing times collected with an experimental structure library containing approx. 880 chemical structures, and extrapolates the results to a production system. A chapter on conceivable further applications of the algorithms developed in this thesis is

adjoined, and - after the summary of the major results - a glossary of terms and a sample search are appended.

The chapters on the data structure and on algorithms require some knowledge of discrete mathematics. The chapter on the implementation assumes familiarity with some computer science terms. For an appreciation of the data structure chosen, an understanding of chemical structure concepts is necessary.

1.3. Current Structure Retrieval Systems

Chemical Abstracts Service (CAS for short) provides access to over 7 million chemical structures mentioned in the chemical literature. The hardware and software used to handle them is described in many publications and manuals [5-10]. I also have personal experience with the interactive substructure search system CAS-Online as it is the main system used in the literature search service at the University of Zürich. Therefore, CAS-Online became the yardstick in the evaluation of GM-Search. The search system of CAS-Online can access only the constitution of the structures, while their configuration (if at all recorded) is stored as text descriptors in the name of the compound.

Two other systems are also important: One is the DARC-System [11, 12], which is provided by Telesystemes Questel and manages the same collection of molecular structures as CAS-Online, but with a different access method. Extensions of DARC to handle stereochemical aspects of chemical structures have been published [13-15], but they are not available with Chemical Abstracts structures.

Another common structure retrieval system is the MACCS/REACCS [16-18] group of programs. They are mainly used to administer "in-house" collections of chemical structures (MACCS) and reactions (REACCS) together

with other data, e.g. literature references, and physical and biological data. REACCS [16] also provides access to (commercially available) collections of information on chemical reactions. The mainframe versions of these systems are also able to handle carbon stereochemistry, but the corresponding algorithms are not published.

2. The Users View of the System

This chapter describes GM-Search as it is seen by the user, who adds chemical structures to the library, or poses substructure queries to the search program.

2.1. User Interface

Graphical interfaces for the input of chemical structures are already very common. They are used in programs for computer aided synthesis planning to input target molecules, in the field of molecular modelling to construct three-dimensional models of enzymes, substrates, and inhibitors for later processing, and in database systems for chemical applications.

At Chemical Abstracts, the structures referred to in an abstracted publication are also entered graphically. The system tests whether structures with the same constitution are already in the database. Then a chemist must decide if the configuration of the processed structure is not already stored [9], and if this is true, the structure is entered into the so-called Registry File. Together with the structure, a name (in most cases also derived by computer) is stored; the chemist then augments this name with text descriptors for the stereochemical aspects of the molecule.

The CAS-Online user can search the Registry File by full-structure, by substructure, or by constituents of the chemical names of the molecules. The result is a set of so-called registry numbers, which can then be used in other files of CAS-Online to retrieve the relevant literature references.

Depending on available hardware, the structure-type queries can be input either graphically or by textual commands. In the latter case, the constructed queries can be displayed on the graphics terminal (if available) or as a character drawing. The same display facilities exist to show the retrieved structures.

2.2. Graphical Input

In GM-Search, the graphical input is handled by the FRAME [19] program. The DRAWEDITOR procedure to input structures is intended to be self-explanatory. Only those commands and options available to input structures and substructures will be described here.

The main input device is a magnetostrictive tablet with a so-called puck. The combination puck/tablet is similar to a mouse but allows the measurement of absolute coordinates instead of relative movements as with the mouse. The position of the puck on the tablet is reflected by the position of the cursor on the screen and the corresponding screen coordinates are then accessible to the DRAWEDITOR.

The basic command mode is the DRAW mode. In DRAW mode the atoms and the bonds between them can be "drawn" and erased using the puck and its four buttons. Double and triple bonds are drawn by entering the same bond two or three times. The command ATOM TYPES is used to specify the atom types of the non-carbon atoms. CHARGE & RADICAL is used to attribute charges or radical state to an atom. With GET DRAWING and ADD COMPONENT predefined structures or structure parts (previously stored by PUT DRAWING) can be read from floppy disk. The stereochemical features of a molecule are input by SPECIAL SYM., which is also used to set aromatic and generic bond types.

A generic bond connects two atoms, but the order (single, double, ...) of this bond is left unspecified. This feature is useful when defining certain generalized substructure queries.

Besides the generic bond type, five special atom types are also used for substructures. The atom type R shows a "free valence" of the substructure. Each of the other four atom type symbols denotes a set of atom types. X stands for a halogen atoms, M for any metal, Q for any non-hydrogen, non-carbon atom, and A for any non-hydrogen atom.

When augmenting the library, the molecule defined with DRAWEDITOR is returned to the main program FRAME and subjected to the TO LIBRARY command. An accompanying line of text is requested from the user, and a text description of the structure is added to a set of intermediate files. A batch of such descriptions can then be added to the library by a separate program. This division was necessary because the task switch from the PASCAL program FRAME to the C-written library updating program is rather time consuming. Thus the library updating is done once for a collection of molecules and not once for each molecule. The library updating program checks for each molecule if it is already stored in the database and adds it only if this is not the case.

A substructure search is initiated by issuing the command SEARCH LIB. A text description of the query is generated and transferred to the library searching program. While the substructure search is performed, the text lines associated with the structures matching the query are shown. The SEARCH LIB. command is left by entering a final <cr>, and the drawing of the query is redisplayed for further editing.

2.3. Contents of the Database

The structures included in the database could not be taken from an existing database but had to be input using the DRAWEDITOR. Therefore, their number was limited to about 1000 structures. Most of them were drawn from [20], which contains representative examples from the "Beilstein Handbook of Organic Chemistry". Another major source of compounds was [21]. Part of a catalogue of commercially available optically active compounds [22] was also included.

The information stored together with the chemical structure was confined to the coordinates of the input drawing and one line of text. The text line is used to store the chemical name, the registry number, and/or the

source of the structure. This is similar to the information in the CAS-On-line Registry file (structure, drawing, registry number, systematic CA-name, and up to 10 references).

3. The Relational System

The aim of this thesis was to show that relational systems can be used as an efficient tool to represent chemical structures within a computer. It is shown in [2] and [3] that chemical structures can be described consistently by augmented relational systems called M-descriptions. The procedure by which M-descriptions are derived is called the M-method. The convenient representation of substructures poses additional problems, which have been tackled by extending the concept of M-descriptions to the so-called generalized M-descriptions or GM-descriptions.

To facilitate the discussion of GM-descriptions, the following chapter summarizes the use of M-descriptions to describe molecular structures (in the following just called molecules) by means of an example.

3.1. M-Descriptions

The term relational system can be defined as follows [4]:

Definition: Let X be a finite set and let $R_1, R_2, \dots, R_m$ be relations of $k_1, k_2, \dots, k_m$ arguments in X , respectively, i.e. R_i is a subset of X^{k_i} for $1 \leq i \leq m$. The ordered $(m+1)$ -tuple

$$S = (X, R_1, R_2, \dots, R_m)$$

is called a finite relational system over the domain X .

In this thesis, relational system is always restricted to finite relational system, and X is a set of consecutive numbers taken from N_0 , the set of natural numbers including zero.

M-descriptions describe molecules by denoting the mobility restrictions particular to the atoms comprising the molecular structure. The mathematical form of this notation is defined as follows.

Definition:

An M-description consists of a relational system $S = (X, R_1, \dots, R_m)$, and a function $f : \{R_1, \dots, R_m\} \rightarrow \{a_1, \dots, a_m\}$ associating a so-called attribute¹⁾ a_i with each relation R_i [2]²⁾.

The domain X contains the numbers used to number the atoms of the molecule being described, and the attributes represent the chemical meaning of the corresponding relations.

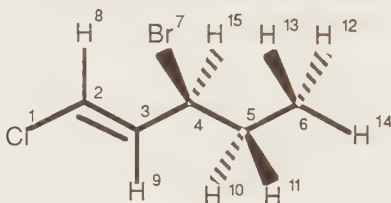


Fig. 3-1

If (2E,4S)-1-chloro-3-bromo-1-pentene (Fig. 3-1) is described with the modified M-Method [3], the following M-description results:

I: I(Br): 7;
 I(Cl): 1;
 I(C): 2, 3, 4, 5, 6;
 I(H): 8, 9, 10, 11, 12, 13, 14, 15;
 C: C(*): 1 2, 2 3, 2 8, 3 4, 3 9, 4 5, 4 7, 4 15,

¹⁾ In [2], they are called predicates as in [54]. Attribute is used here as a less misleading term.

²⁾ In this thesis sets will be designated by upper case letters while lower case will be used for scalar values, and tuples of elements from some set.

5 10, 5 11, 5 6, 6 12, 6 13, 6 14;

D: D(0): 4 8, 1 9;

D(180): 1 4, 8 9;

O: O(+): 3 5 15 7, 5 12 13 14, 4 6 11 10.

The relation attributes are I(Br), I(Cl), ..., O(+). The tuples comprising the relations are listed following those attributes and separated by commas. The relations are grouped into blocks of similar meaning and equal tuple size. The blocks are named I-block for identification, C-block for connectivity, A-block³⁾ for angularity, D-block for dihedrality, and O-block for orientivity⁴⁾.

This M-description should be read as follows:

I(Br): 7 Atom 7 is a bromine atom.

I(Cl): 1 Atom 1 is a chlorine atom.

...

C(*): 1 2, Atom 1 is connected to atom 2,
2 3,... atom 2 is connected to atom 3....

D(0°): 4 8, 1 9 The atom pairs (4,8) and (1,9) describe a
distance which is quasifixed because of a
dihedral angle of approx. 0°.

...

O(+): 3 5 15 7, The locations of the atoms 3, 5, 15, and 7
lie on a right-handed spiral.

3) The A-block was dropped for reasons of chemical redundancy as described below.

4) The term orientivity was coined in our group for this relation to go with connectivity, angularity, and dihedrality. Orientation is used for particular tuples from the orientivity relation.

The M-description was shortened as prescribed by [3]. This is done mainly by suppression of chemically redundant information:

- The bond orders have been neglected, because (for the ground state) this information can be derived from the valence and ligance of the atoms involved together with the dihedrality relation.
- Similarly, the description of the bond angles can often be inferred from the ligancy.
- The dihedral angle tuples (and the bond angle tuples, if not redundant) are interpreted as fixing distances, and therefore, only the first and the last atoms are mentioned.

Another means of shortening is the suppression of geometric redundancy:

- Because the distance function is symmetric, the distances described by the C-, A-, and D-blocks are specified only by one tuple each.
- From four ligands being on a right-handed spiral, it can be concluded that any even permutation of the four centers involved also lies on a right-handed spiral. Hence, the orientivities of the ligands around the atoms 4, 5, and 6 are described by only one O-tuple for each center.

The chemical structure of the molecule can be derived uniquely from its M-description. On the other hand, one molecule has many M-descriptions because the numbering of the atoms is arbitrary. However, given a numbering, the tuples to be included in the description can be selected in a unique way [2]. The ambiguity in numbering is resolved by a process called canonization, which yields one canonical numbering independent from the input numbering. One way to select a canonical numbering is to use the one which gives the lexicographically smallest M-description, if the M-description is regarded as a string of numbers [2,4].

It is easy to see, that this rule selects a unique⁵⁾ numbering from the possible ones. However, in general this "minimum method" [4] is laborious. The chapter on canonization will show that this approach can be improved using a refinement procedure and a method to handle highly symmetric structures.

The unique M-description of a molecule can then be used as an identification key in a storage and retrieval system for chemical structures.

The preceding discussion shows how to describe an organic molecule by an M-description. For further examples, especially on the description of non-carbon stereochemistry, see [2].

3.2. Generalized M-Descriptions

M-descriptions have some properties that make them especially suitable for computer storage of chemical information:

- They describe a molecule uniquely.
- They are easily translated to a molecular structure and vice-versa.
- They are concise.
- They describe all information (atom types, bonding, stereochemical aspects) contained in the structure in small units, and these units of information are handled homogeneously.

⁵⁾ up to automorphisms

The first three properties are important to any storage and retrieval system. The last one makes M-descriptions attractive as tools for substructure search systems (SS-systems), especially if stereochemical substructures have to be retrieved.

On the other hand, some features of the M-method are not desirable for an SS-system:

- The actual bond order of a bond between two atoms is supposed to be derivable from the context of the description, but the context might be unknown in the description of a substructure.
- Angularity and dihedrality relations are described without the central one or two atoms, respectively. This prevents, for instance, a distinction between the two $D(0^\circ)$ relationships (1 2 3 4) and (1 6 5 4) in 1,4-cyclohexadiene (Fig. 3-2)

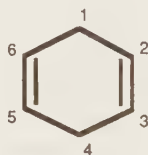


Fig. 3-2

- The relation attributes describe only one property. For defining substructure queries, it should be possible to express a set of alternative properties⁶⁾, i.e. a generic property.

⁶⁾ E.g., a set of permissible atom or bond types.

Also some technical features of the data structure of M-descriptions hinder an efficient implementation of the concept:

- The omission of the geometrically redundant permutations of the relation tuples in the shortened description would make it necessary, to expand the structure description before performing a substructure test.
- The information about a certain center is distributed among many tuples in the different relations.

These arguments lead to some definitions and to a computer implementation strategy:

Definition:

Let G be a permutation group acting on k -tuples of elements from some set X by permuting the positions of the tuple⁷⁾. Two k -tuples t_1 , t_2 are called G -equivalent, if there exists a permutation $g \in G$ with $t_2 = g(t_1)$.

Definition:

Given

X , a finite set called the domain,

BD , a finite set of so-called block descriptors⁸⁾,

7) This is an informal formulation of a permutation group acting on the index set of the k -tuples and the derived action on the k -tuples. They should not be confused with possible permutation groups acting on X , for which an action on tuples of elements from X could also be derived, but which are not meant here.

FG, a function mapping BD into the set of permutation groups acting on tuples of elements from X, and

AS, a finite set of attributes.

Let U_0 be the set of all triples $u=(b, A, t)$, called units, where

$b \in BD$ with $FG(b)$ acting on k -tuples of elements from X,

A is a non-empty subset of AS, and

t is a k -tuple of elements of X.

The quadruple $Q=(X, FG, AS, U)$ with U being a subset of U_0 is called a generalized M-description, or GM-description, if there are no two different units $u_1=(b_1, A_1, t_1)$, $u_2=(b_2, A_2, t_2) \in U$ for which

- $b_1 = b_2$ ($= b$),
- the intersection of A_1 and A_2 is not empty, and
- t_2 is $FG(b)$ -equivalent to t_1 ⁹).

As in the case of M-descriptions, without loss of generality, the domain X can be a subset of N_0 used to number the atoms of the molecule or sub-structure.

It is easy to see that M-descriptions can be expressed as generalized M-descriptions by setting $FG(b) = \{e\}$ for all $b \in BD$ and using singleton sets as attribute sets.

⁸) They represent the blocks, mentioned in the informal discussion on the chemical interpretation of M-descriptions.

⁹) This has the effect that a) only one of the permutationally equivalent units is actually included, and b) units with the same tuple do not interfere in the description of a property. A similar restriction distinguishes undirected graphs from multigraphs.

The following definitions designate the congruence relation on GM-descriptions.

Definitions:

Let t be a tuple of numbers and let the permutation group G act on t . The lexicographically smallest tuple $g(t)$ with $g \in G$ is called the G -minimum of t or $\min_G(t)$.

The permutational minimum of a unit $u=(b, A, t)$ is the unit $\min(u)=(b, A, \min_{FG(b)}(t))$. Two units $u_1=(b_1, A_1, t_1)$ and $u_2=(b_2, A_2, t_2)$ are called congruent if $\min(u_1)=\min(u_2)$.

Two GM-descriptions $Q_1=(X_1, FG_1, AS_1, U_1)$ and $Q_2=(X_2, FG_2, AS_2, U_2)$ are regarded as congruent (written $Q_1 \equiv Q_2$), if $X_1=X_2$, $FG_1=FG_2$, $AS_1=AS_2$, and the sets of permutational minima of their units are equal.

In a given SS-system the set of possible B-values and the function FG are fixed. For the presented SS-system, the blocks, the corresponding permutation groups, and the basic¹⁰⁾ attributes constituting the attribute sets are summarized in Table III-1.

In the computer implementation, all units are stored using the same data structure with the following components:

- The unit block descriptor, which by means of the function FG also specifies the permutation group and the tuple size.

¹⁰⁾ In GM-Search, the actual SS-system discussed here, the structures have been pruned and preprocessed to improve storage efficiency and running time. This results in additional attributes in the atom and bond attribute sets.

Table III-1

blocks $b \in BD$	attributes $a \in AS$	groups $G = FG(b)$
identification or IDENT	atom type: H, He.... charge: plus4, plus3, ..., minus4 radical state: radical	$\{(1)\} = S_1$
connectivity or BOND	single, double, triple, aromatic, higher	$\{(1)(2), (1\ 2)\} = S_2$
angularity or ANGLE	a060, a090, a109, a120, a180	$\{(1)(2)(3), (1\ 3)(2)\}$
dihedrality or DIHED	d00 , d060, d090, d120, d180	$\{(1)(2)(3)(4), (1\ 4)(2\ 3)\}$
orientivity or ORIENT	plus	$\{(1)(2)(3)(4), (1\ 2\ 3)(4),$ $\dots\} = A_4$

- A bit string or an index into a table of bit strings to represent the attribute set.

- An array of center numbers specifying the atoms contained in the tuple of the unit.

The domain of the tuples of the units is a set of consecutive numbers from N_0 .

As prescribed by the definition, for each permutational equivalence class of units only the lexicographically smallest one (i.e. the permutationally minimal one) is stored, and the permutation group is directly used in the

algorithms for canonization and substructure matching. The problem of the information being spread out over the relations is solved by double book-keeping; i.e. for each center, a cross reference list of the units containing that center is kept.

According to these explanations a GM-description for the molecule of Fig. 3-1 would read as follows:

Units:

<u>1</u> :	IDENT	Br	7
<u>2</u> :	IDENT	Cl	1
<u>3</u> :	IDENT	C	2
<u>4</u> :	IDENT	C	3
<u>5</u> :	IDENT	C	4
<u>6</u> :	IDENT	C	5
<u>7</u> :	IDENT	C	6
<u>8</u> :	IDENT	H	8
<u>9</u> :	IDENT	H	9
<u>10</u> :	IDENT	H	10
<u>11</u> :	IDENT	H	11
<u>12</u> :	IDENT	H	12
<u>13</u> :	IDENT	H	13
<u>14</u> :	IDENT	H	14
<u>15</u> :	IDENT	H	15
<u>16</u> :	BOND	double	2 3
<u>17</u> :	BOND	single	1 2
<u>18</u> :	BOND	single	2 8
<u>19</u> :	BOND	single	3 4
<u>20</u> :	BOND	single	3 9
<u>21</u> :	BOND	single	4 5
<u>22</u> :	BOND	single	4 7
<u>23</u> :	BOND	single	4 15
<u>24</u> :	BOND	single	5 6
<u>25</u> :	BOND	single	5 10

<u>26</u> :	BOND	single	5	11
<u>27</u> :	BOND	single	6	12
<u>28</u> :	BOND	single	6	13
<u>29</u> :	BOND	single	6	14
<u>30</u> :	DIHED	d00	1	2 3 9
<u>31</u> :	DIHED	d00	4	3 2 8
<u>32</u> :	DIHED	d180	1	2 3 4
<u>33</u> :	DIHED	d180	8	2 3 9
<u>34</u> :	ORIENT	plus	3	5 15 7
<u>35</u> :	ORIENT	plus	5	12 13 14
<u>36</u> :	ORIENT	plus	4	6 11 10

Cross reference:

- 1) 2, 17, 30, 32
- 2) 3, 16, 17, 18, 30, 31, 32, 33
- 3) 4, 16, 19, 20, 30, 31, 32, 33, 34
- 4) 5, 19, 21, 22, 23, 31, 32, 36
- 5) 6, 21, 24, 25, 26, 34, 35
- 6) 7, 24, 27, 28, 29, 36
- 7) 1, 22, 34
- 8) 8, 18, 30, 31, 33
- 9) 9, 20, 33
- 10) 10, 25, 36
- 11) 11, 26, 36
- 12) 12, 27, 35
- 13) 13, 28, 35
- 14) 14, 29, 35
- 15) 15, 23, 34

In the following the (redundant) cross reference part will only be given if it is necessary to illustrate a certain point or helpful to work through an example. Each unit line begins with an identifier specifying the unit block

followed by a list of identifiers for the attribute set¹¹⁾ and a list of numbers for the atoms involved.

Although this text description is not short, the actual storage requirement of the compressed binary description is small.

Definition:

A GM-description $Q_1=(X_1, FG_1, AS_1, U_1)$ is a sub-GM-description of $Q_2=(X_2, FG_2, AS_2, U_2)$ (written $Q_1 \leq Q_2$) if $FG_1 = FG_2$ and $AS_1 = AS_2$, and if there exist injective mappings $f_X: X_1 \rightarrow X_2$ and $f_U: U_1 \rightarrow U_2$, such that

for every $(b_1, A_1, t_1) \in U_1$ with $f_U((b_1, A_1, t_1)) = (b_2, A_2, t_2)$ holds

- a) $FG_1(b_1) = FG_2(b_2) = G$,
- b) A_1 is a subset of A_2 , and
- c) there is a $g \in G$ with $g(t_2) = f_X(t_1)$.

If f_X is known, f_U is implied by means of the final restriction in the definition of GM-descriptions. Fig. 3-3 shows a substructure contained in the molecule of Fig. 3-1.

Besides the free valences (drawn as unnumbered "R-atoms") it also contains the generic atom 4, which is labelled by a set of permitted atom types, namely chlorine, bromine, and iodine. The text description is shown below:

¹¹⁾ If a molecular structure is described, those lists initially consist of only one element. As mentioned before, for atom and bond type sets additional attributes are define and they are included after further processing. See also in the chapters on the pruning and ring size algorithms.

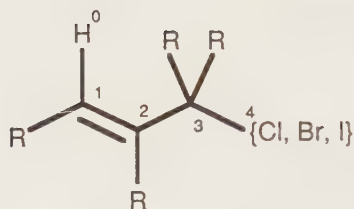


Fig. 3-3

Units:

<u>0</u> :	IDENT	Cl Br I	4
<u>1</u> :	IDENT	C	1
<u>2</u> :	IDENT	C	2
<u>3</u> :	IDENT	C	3
<u>4</u> :	IDENT	H	0
<u>5</u> :	BOND	double	1 2
<u>6</u> :	BOND	single	0 1
<u>7</u> :	BOND	single	2 3
<u>8</u> :	BOND	single	3 4
<u>9</u> :	DIHED	d00	0 1 2 3

The domain mapping f_X and the unit mapping f_U of the GM-description of the substructure of Fig. 3-3 on the one of the molecule of Fig. 3-1 is:

$$f_X: \{0 \rightarrow 8, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 7\},$$

$$f_U: \{0 \rightarrow 1, 1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 5, 4 \rightarrow 8, 5 \rightarrow 16, 6 \rightarrow 18, 7 \rightarrow 19, 8 \rightarrow 22, 9 \rightarrow 31\}.$$

In the structure of the last example, the atom numbering started with 0. This was done for easy implementation in the programming language C and will be maintained in the following.

4. Algorithms

The previous chapter described the concept of GM-descriptions and their use to represent molecular structures and substructures. Based upon GM-descriptions, the structure retrieval system GM-Search has to perform the following operations:

- Add a GM-description to the library of structures.
- Given a GM-description of a substructure query, retrieve all structures in the library containing the query as sub-GM-description.

Adding a structure to the library means pruning (of hydrogen atoms), preprocessing (to speed up further processing), canonization, and storage of unique GM-descriptions. Searching for structures containing a substructure query is done by screening out candidates for containing the query as sub-GM-description, and subsequent sub-GM-testing.

This chapter will present the key algorithms used in GM-Search in more detail.

4.1. Structure Pruning

On the average, about half of the atoms in an organic molecule are monovalent hydrogen atoms. Therefore, virtually all current chemical structure database systems store hydrogen atoms in a special way.

The CAS-Online system, for instance, handles the number of hydrogen atoms attached to a non-hydrogen atom as an attribute of that atom. Most organic synthesis planning programs also apply this procedure. The process of transforming bonded hydrogens to their attribute representation is called pruning.

Pruning is applied because the computing time of many algorithms manipulating molecule representations grows with the number of atoms and with the number of symmetry elements of the structures¹²). Another advantage of pruning is related to substructure testing: The ligancy of a carbon atom (Besides hydrogen, carbon is the most frequent atom in organic molecules) is determined by the orders of its bonds, whereas the non-hydrogen ligancy is an additional graph invariant, which can be exploited to classify atoms in the substructure testing algorithm to be described later. Treating hydrogen atoms in a special way results in a "look-ahead" for this algorithm because part of the neighborhood information is then directly used to distinguish atoms. For these reasons, it is usually beneficial to describe hydrogen atoms in a special manner, so that they can be neglected for many operations. For systems not including stereochemical aspects, pruning is an easy task because the X-H bond can be reconstructed from the attribute information stored with the X-atom.

However, hydrogen atoms may on occasion carry essential stereochemical information. Thus the approach used here utilizes hydrogen count attributes in the form of special atom-type sets and retains those hydrogen atoms that carry stereochemical information.

Some situations that can arise for the case of X being a carbon atom are shown in Fig. 4-1.

The rules applied for pruning are:

- For each IDENT unit describing the atom type set of an atom X, include one of the attributes H0, H1, H2, H3, or H4 corresponding to the number of hydrogen neighbors of X in the attribute set.

¹²) Each methyl group (CH_3) gives rise to an additional symmetry element describing its internal rotation.

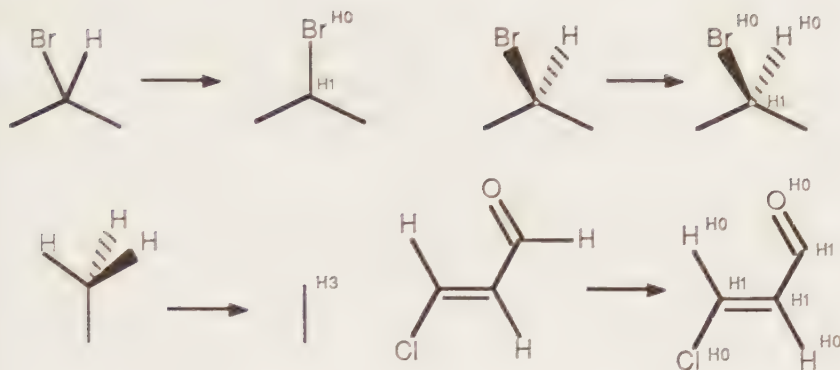


Fig. 4-1

- Retain all hydrogen atoms that do not have exactly one non-hydrogen neighbor.
- Retain all hydrogen atoms, which are members of an ANGLE- or DIHED-unit.
- Retain all hydrogens which are members of some ORIENT-unit if there is no other pruneable hydrogen in the same ORIENT-unit connected to the same atom.
- Prune away all other hydrogen atoms.

Fig. 4-2 shows the pruned structure of Fig. 2-1. The hydrogen counts are attached to the atoms as labels H0, ..., H3. In the following figures they will usually be defined implicitly and will be specified only if they illustrate a particular aspect of a problem.

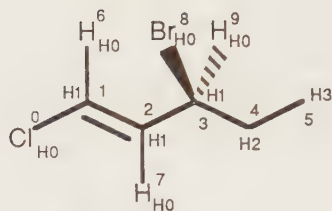


Fig. 4-2

The corresponding GM-description will be:

```

IDENT  Br  H0  8
IDENT  Cl  H0  0
IDENT  C   H3  5
IDENT  C   H2  4
IDENT  C   H1  1
IDENT  C   H1  2
IDENT  C   H1  3
IDENT  H   H0  6
IDENT  H   H0  7
IDENT  H   H0  9
BOND    double  2  1
BOND    single  1  0
BOND    single  3  2
BOND    single  4  3
BOND    single  5  4
BOND    single  6  1
BOND    single  7  2
BOND    single  8  3
BOND    single  9  3
DIHED   d180   3  2  1  0
DIHED   d180   7  1  2  6
DIHED   d00    3  2  1  6
DIHED   d00    7  2  1  0
ORIENT  plus   9  2  4  8
  
```

For pruning full structures, it is sufficient to add one hydrogen count attribute to the IDENT attribute set for each atom. Substructure pruning is more involved. Therefore, the first rule for structure pruning must be extended as follows:

- For each IDENT-unit describing the atom type set of an atom X add the range $H_n, \dots, H_m$ of hydrogen count attributes to the atom type set. n is the minimum number of hydrogen atoms attached to X (i.e., the actual number of hydrogen atoms bonded to X in the query), and m is their maximum number (i.e., the number of hydrogen atoms plus the number of "R-atoms").

With that rule, the pruned structure of Fig. 2-3 is written as:

IDENT	Cl	Br	I	H0	4
IDENT	C	H1	H2		1
IDENT	C	H0	H1		2
IDENT	C	H0	H1	H2	3
IDENT	H	H0			0
BOND	double				1 2
BOND	single				0 1
BOND	single				2 3
BOND	single				3 4
DIHED	d00				0 1 2 3

This contraction of hydrogen counts into the atom type attribute sets also has a disadvantage:

Generic queries which include hydrogen in the atom type set of a query atom are no longer meaningful and must be replaced by an "OR" combination of two, or more, standard queries. This drawback, however, is not peculiar to GM-descriptions; it seems to be a general problem of pruning.

4.2. Preprocessing

GM-search is capable of performing a substructure search. During substructure search, an atom-by-atom matching process is used to establish a unit-preserving mapping of the atoms of the query onto the atoms of the molecule. If this process is guided only by information about the local environment of the atoms, it is time-consuming to exclude mappings between the atoms of rings of different sizes. To avoid this, the GM-descriptions of the molecules and the queries are preprocessed to collect ring size information. For each BOND-unit¹³⁾ (either in the GM-description of the molecule or the query), the size of the smallest ring containing that bond is computed¹⁴⁾. This minimal ring size information is stored as additional BOND attributes. The attributes are 'r3b', 'r4b', ..., 'r8b', or 'rnb' for rings with three, four, ..., eight, or more members, respectively, and 'chainb' for chain bonds. The example of Fig. 4-3 shows how they are used:

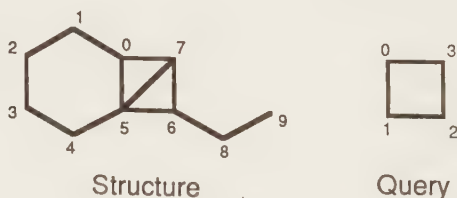


Fig. 4-3

¹³⁾ In the following, BOND-unit is abbreviated to "bond" if it is unambiguous.

¹⁴⁾ The GM-description is transformed into a graph, which does not contain loops and double edges for chemical structures.

The structure contains, with respect to minimal ring size, three classes of bonds. In the structure the bonds (0 1), (1 2), (2 3), (3 4), and (4 5) are r6b-bonds, whereas (0 5), (0 7), (6 7), (5 6), and (5 7) are r3b-bonds. The bonds (6 8) and (8 9) are chainb-bonds. All the query bonds are of type {r3b, r4b}¹⁵⁾. A query bond a can only be mapped on a structure bond if the minimal ring size of a is larger than, or equal to, that of b. Just by comparing the bond attributes it can now be concluded that the four bonds of the query could only be mapped on those five bonds of the structure which bear an r3b attribute, whereas without this information all the bonds except (8 9) would be candidates.

4.2.1. Computing Ring Size Information

Moore [25] developed an algorithm to compute the shortest path between two vertices of a graph by breadth-first search. Its basic idea was apparently reinvented in a chemical context by Bersohn [26]. These algorithms have been extended in this work to find, for each edge of the graph, the size of the smallest ring that it is part of. This section will give a pseudo-code description of this algorithm, in sufficient detail to enable its implementation in any programming language.

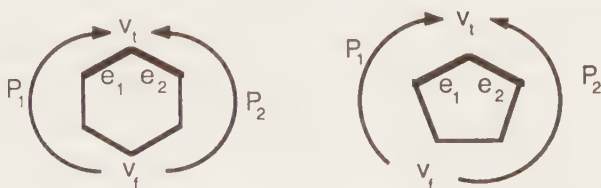


Fig. 4-4

¹⁵⁾ The computed ring size attributes of query bonds have to be extended by also adding the attributes for all smaller ring sizes.

The algorithm exploits the following fact (see Fig.4-4): If two edges e_1 , e_2 , joined at a common vertex v_t , are part of a ring of size s , there must be a vertex v_f with the following properties:

- There are two paths P_1 , P_2 from v_f to v_t with e_1 and e_2 as their last edges, respectively, and
- $\text{length}(P_1) + \text{length}(P_2) = s$, and $|\text{length}(P_1) - \text{length}(P_2)| \leq 1$.

If s is the size of the smallest ring containing both e_1 and e_2 , then P_1 and P_2 are edge-disjoint, and there is no shorter path joining v_f with v_t having e_1 or e_2 as last edges, respectively.

This means: If, for each triple (v_f, v_t, e) ($v_f \neq v_t$, and $v_t \in e$), we examine one of the possibly many shortest paths from v_f to v_t with e as terminal edge, we can compute the minimal size of the rings, which contain a certain bond. Because only the two cases of Fig. 4-4 must be considered, it suffices to examine only those triples where the length of the corresponding shortest paths are bounded by the graph theoretical distance of v_f and v_t plus 1.

Furthermore, if we generate those paths in a breadth-first manner, then for each pair (v_f, v_t) only the edge of the first triple, denoted $(v_f, v_t, e)'$, must be stored. The ring sizes for all other edges can now be directly computed as they are encountered, while the ring size for the stored edge is computed using all other edges meeting at v_t .

Ring Perception Algorithm:

Declaration:

Given

$G=G(V,E)$: a graph with vertex set V and edge set E .

D: $|V| \times |V|$ -matrix to store the lengths of the shortest paths between two vertices,
T: $|V| \times |V|$ -matrix to store the last edges of the shortest paths,
R: $|E|$ -vector to store for each edge the size of the smallest ring containing the edge, and
openpaths,
newpaths: sets of triples (v_f, v_t, e) ; the path denoted by a triple runs from v_f to v_t and e is the last edge of this path.

Initialization:

```
D[v1, v2] ← 0 for all v1, v2 ∈ V;
R[e] ← ∞ for all e ∈ E;
openpaths ← {};
for all e = e(v1, v2) ∈ E
    openpaths ← openpaths ∪ {(v1, v2, e), (v2, v1, e)};
endfor;
ndist ← 1;
```

Computing paths and rings:

/* Loop invariant:

If the shortest path between the vertices v_f and v_t has a length $n < \text{ndist}$, then $D[v_f, v_t] = n$ and $T[v_f, v_t]$ is terminal edge of the first encountered shortest path from v_f to v_t .

If the minimal ring size of a ring containing a bond e is $s < 2 * \text{ndist}$, then $R[e] = s$.

If there is a shortest path of length $< \text{ndist}$ from v_f over e as last edge to v_t , and if the distance of v_f and v_t is $\geq \text{ndist} - 1$, then openpaths contains a triple (v_f, v_t, e) .

*/

```
while openpaths ≠ {} do
    newpaths ← {};
    for all triples (vf, vt, e) ∈ openpaths do
        if D[vf, vt] = 0 then          /* No ring closure */
```

```

D[vf,vt] ← ndist;
T[vf,vt] ← e;
for all neighbors vn of vt which are not in e
    newpaths ← newpaths ∪ (vf, vn, (vt,vn));
endfor;
else
    R[e] ← min(R[e],ndist+D[vf,vt]);
    R[T[vf,vt]] ← min(R[T[vf,vt]],ndist+D[vf,vt]);
endif;
endfor;
ndist ← ndist+1;
openpaths ← newpaths;
endwhile;

```

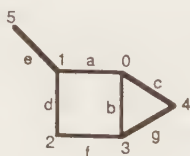


Fig. 4-5

The graph of Fig. 4-5 is used as an example to illustrate the flow of the algorithm:

$G = G(V,E);$

$V = \{0, 1, 2, 3, 4, 5\},$

$E = \{a=(0\ 1),\ b=(0\ 3),\ c=(0\ 4),\ d=(1\ 2),\ e=(1\ 5),\ f=(2\ 3),\ g=(3\ 4)\}.$

The contents of the variables are shown when all assignments to a variable have been done during one while-cycle. The character '.' in the array T means 'not defined'.

D:	0	1	2	3	4	5	T:	0	1	2	3	4	5	R:	a	:	∞
0	0	0	0	0	0	0		0	.	.	.	.	.		b	:	∞
1	0	0	0	0	0	0		1	.	.	.	.	.		c	:	∞
2	0	0	0	0	0	0		2	.	.	.	.	.		d	:	∞
3	0	0	0	0	0	0		3	.	.	.	.	.		e	:	∞
4	0	0	0	0	0	0		4	.	.	.	.	.		f	:	∞
5	0	0	0	0	0	0		5	.	.	.	.	.		g	:	∞

openpaths: (0,1,a), (1,0,a), (0,3,b), (3,0,b), (0,4,c), (4,0,c),
 (1,2,d), (2,1,d), (1,5,e), (5,1,e), (2,3,f), (3,2,f),
 (3,4,g), (4,3,g)

ndist: 1

D:	0	1	2	3	4	5	T:	0	1	2	3	4	5	R:	a	:	∞	
0	0	1	0	1	1	0		0	.	a	.	b	c	.		b	:	∞
1	1	0	1	0	0	1		1	a	.	d	.	.	e		c	:	∞
2	0	1	0	1	0	0		2	.	d	.	f	.	.		d	:	∞
3	1	0	1	0	1	0		3	b	.	f	.	g	.		e	:	∞
4	1	0	0	1	0	0		4	c	.	.	g	.	.		f	:	∞
5	0	1	0	0	0	0		5	.	e	.	.	.	.		g	:	∞

openpaths: (0,5,e), (0,2,d), (1,4,c), (1,3,b), (0,2,f), (0,4,g),
 (3,1,a), (3,4,c), (0,3,g), (4,1,a), (4,3,b), (1,3,f),
 (2,0,a), (2,5,e), (5,0,a), (5,2,d), (2,0,b), (2,4,g),
 (3,1,d), (3,0,c), (4,0,b), (4,2,f)

ndist: 2

D:	0	1	2	3	4	5	T:	0	1	2	3	4	5	R:	a	:	4	
0	0	1	2	1	1	2		0	.	a	d	b	c	e		b	:	3
1	1	0	1	2	2	1		1	a	.	d	b	c	e		c	:	3
2	2	1	0	1	2	2		2	a	d	.	f	g	e		d	:	4
3	1	2	1	0	1	0		3	b	a	f	.	g	.		e	:	∞
4	1	2	2	1	0	0		4	c	a	f	g	.	.		f	:	4
5	2	1	2	0	0	0		5	a	e	d	.	.	.		g	:	3

openpaths: (0,3,f), (1,3,g), (1,2,f), (1,4,g), (3,2,d), (3,5,e),
 (4,2,d), (4,5,e), (2,3,b), (2,4,c), (5,4,c), (5,3,b),
 (5,3,f), (2,0,c), (4,1,d)

ndist: 3

D:	0	1	2	3	4	5	T:	0	1	2	3	4	5	R:	a	:	4	
0	0	1	2	1	1	2		0	.	a	d	b	c	e		b	:	3
1	1	0	1	2	2	1		1	a	.	d	b	c	e		c	:	3
2	2	1	0	1	2	2		2	a	d	.	f	g	e		d	:	4
3	1	2	1	0	1	3		3	b	a	f	.	g	e		e	:	∞
4	1	2	2	1	0	3		4	c	a	f	g	.	e		f	:	4
5	2	1	2	3	3	0		5	a	e	d	b	c	.		g	:	3

openpaths: (5,3,g), (5,2,f), (5,4,g)

ndist: 4

The last iteration empties openpaths and so the final result is:

- The bond b, c, and g are part of a three membered ring (besides possible larger rings).

- a, d, and f are part of a four membered ring.

- e is a chain bond.

4.3. Canonizing

From the outset of non-numeric computer applications in chemistry, testing whether or not two computer representations describe the same molecule has been of central importance. When (chromatic [11]) graph representations (also called connection tables) are used, the problem can be reduced to the graph isomorphism problem (see for instance [27] and the references cited therein).

In most chemical applications however, the problem is not only the comparison of two representations but the comparison of one representation with a whole set of others such as: a database of known compounds (CAS-On-line), readily available precursors, synthetic intermediates (computer assisted synthesis planning), or structure candidates fulfilling certain spectroscopically derived constraints (structure elucidation).

The one-to-one testing of isomorphism of graphs can be done in polynomial time¹⁶⁾, if the valence or degree of the vertices is bounded [28]. However, the time needed to perform this algorithm increases with the 4-th power of the number of vertices in the worst case [45], even if only tri-valent graphs are considered. Fortunately the difficult cases, which make the worst case behavior of the graph theoretical algorithm so bad, only rarely occur in chemistry. Nevertheless, isomorphism testing is generally time-consuming.

¹⁶⁾ I.e. the running time of the algorithm is bounded by a polynomial in the size of the input.

Because in chemistry the problem is of the one-against-many type, the approach of unique coding¹⁷⁾ is used. Once unique codes for all representations are computed, the comparisons can be done with a complexity at most linear in the size of the codes derived.

Many authors made contributions to this problem (see [4, 10, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]) All these papers except [4], [38], and [40] deal with the canonical naming of chromatic graphs describing only the constitution of the molecules. The groups of Gelernter [38] and Wipke [40] were working on computer aided synthesis planning, and they considered stereochemical aspects to be crucial in this field. Therefore, they had to develop canonization procedures for codes including stereochemical information.

Their methods to describe molecules were only directed towards carbon stereochemistry and use parity symbols to represent different configurations, although the SEMA algorithm of Wipke's group has been extended to handle penta-coordinated phosphorus atoms, as well [41].

One contribution of this dissertation is an "efficient"¹⁸⁾ algorithm to find a canonical numbering of the centers of a GM-description of a molecule and by that means compute a unique code for this molecule including its stereochemical aspects.

17) Unique naming, canonical naming, canonical coding, or just canonization are also often used.

18) In some of the mathematical literature, an algorithm is said to be efficient only if it operates with a polynomial worst case complexity. I do not claim this property for the algorithm presented here.

The following sections describe how canonical numberings are found if orderings of the classes and attributes of the GM-descriptions are given.

As an additional result the algorithm computes generators of the automorphism groups of GM-descriptions. Those generators are used within the algorithm to partition the centers into symmetry equivalence classes. This information could also be utilized by other algorithms, i.e. for substructure testing, to avoid redundant computation.

4.3.1. Refinement

An obvious algorithm to find a unique code for a GM-description would be to select among the codes for all possible numberings the lexicographically smallest (or largest) one. Even if there are ways to restrict the number of numberings having to be considered for the canonical selection, this approach can be very time-consuming.

4.3.2. Refinement of Graphs

In graph theory (and in chemical database systems based on the graph concept), this efficiency problem is resolved by refinement procedures to reduce the number of candidates for canonical selection [10, 42].

We now define the terms partition, ordered partition, and prenumbering to help in the description of algorithms for refinement and canonical numbering.

Definitions:

Let V be a set. A partition of V is a collection P of disjoint, non-empty subsets of V whose union is V . The elements of P are called classes (or cells [43]{McK78}). An ordered partition of V is a sequence $(C_1, \dots, C_k)$ for which $\{C_1, \dots, C_k\}$ is a partition.

Let $P=(C_1, \dots, C_k)$ be an ordered partition of a set V . A function $f_P: V \rightarrow \{0, \dots, |V|-1\}$ is called the P-prenumbering of V if:

for every $v \in C_j$, $f_P(v)$ is the sum of all $|C_i|$ with $i < j$.

From a given P -prenumbering f_P , the corresponding ordered partition can be computed by $C_i \leftarrow \{v \in V \mid f_P(v)=i-1\}$.

The graph refinement procedures use properties that are invariant under isomorphism to build ordered partitions of the vertices of the graphs. The neighboring relationships are used iteratively to refine the partitions further, i.e. to split the classes into smaller ones. This is possible because an isomorphism mapping x to y will also map the neighborhood of x to the neighborhood of y . Therefore, the resulting partitioning stays isomorphism invariant.

If the graph is not symmetric, the refinement often ends up with each class containing only one vertex and since the partition is ordered, it then defines a canonical numbering of the vertices of the graph.

As an example of a refinement procedure, the following algorithm will be applied to the graph of Fig. 4-6. The labels $L_1, \dots, L_8$ are used as break points in the trace of the algorithm.

Declaration:

Given

$G=G(V,E)$: graph with vertex set V and edge set E

P : ordered partition of V

f_P : prenumbering

H : ordered sequence of sets of vertices

L : $|V|$ -vector of lists of prenumbers

split: flag; TRUE if a subset was split, FALSE otherwise

Initialization:

L1: $P \leftarrow (c_1, \dots, c_l)$, the ordered partition of V
 with one set per degree and ordered by decreasing degree;
 L2: $f_P \leftarrow P$ -prenumbering;

Splitting:

```

      repeat
L3:   H  $\leftarrow$  ();
      split  $\leftarrow$  FALSE;
      for each set  $c_i$  in  $P=(c_1, \dots, c_l)$ 
        if  $|c_i| = 1$ 
          H  $\leftarrow$  H,  $c_i^{19}$ ;
        else
          for each  $v \in c_i$ 
            L[v]  $\leftarrow$  <list of all  $f_P(v_j)$  with  $v_j$  being neighbor of  $v$ >;
            Sort L[v];
          endfor;
L4:   if all list L[v],  $v \in c_i$  are equal
          H  $\leftarrow$  H,  $c_i$ ;
        else
L5:   H  $\leftarrow$  H, <ordered partition of  $c_i$  using lex. comp. of L[v]>;
          split  $\leftarrow$  TRUE;
        endif;
      endif;
    endfor;
L6:   P  $\leftarrow$  H;
L7:    $f_P \leftarrow$  P-prenumbering;
L8:   until not split;
```

19) " , " means concatenation (APL-syntax).

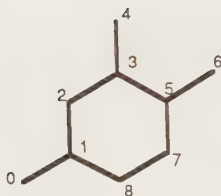


Fig. 4-6

Trace of Algorithm for Example:

L1: $P = (\{1, 3, 5\}, \{2, 7, 8\}, \{0, 4, 6\})$

L2: $f_p = \{1 \rightarrow 0, 3 \rightarrow 0, 5 \rightarrow 0, 2 \rightarrow 3, 7 \rightarrow 3, 8 \rightarrow 3, 0 \rightarrow 6, 4 \rightarrow 6, 6 \rightarrow 6\}$

L3: $H = ()$

L4: $L[1]: 3, 3, 6; L[3]: 0, 3, 6; L[5]: 0, 3, 6;$

L5: $H = (\{3, 5\}, \{1\})$

L4: $L[2]: 0, 0; L[7]: 0, 3; L[8]: 0, 3;$

L5: $H = (\{3, 5\}, \{1\}, \{2\}, \{7, 8\})$

L4: $L[0]: 0; L[4]: 0; L[6]: 0;$

L5: $H = (\{3, 5\}, \{1\}, \{2\}, \{7, 8\}, \{0, 4, 6\})$

L6: $P = (\{3, 5\}, \{1\}, \{2\}, \{7, 8\}, \{0, 4, 6\})$

L7: $f_p = \{3 \rightarrow 0, 5 \rightarrow 0, 1 \rightarrow 2, 2 \rightarrow 3, 7 \rightarrow 4, 8 \rightarrow 4, 0 \rightarrow 6, 4 \rightarrow 6, 6 \rightarrow 6\}$

L8: $\text{split} = \text{TRUE} \rightarrow \text{repeat}$

L3: $H = ()$

L4: $L[3]: 0, 3, 6; L[5]: 0, 4, 6;$

L5: $H = (\{3\}, \{5\})$

L4: $L[7]: 0, 4; L[8]: 2, 4;$

L5: $H = (\{3\}, \{5\}, \{1\}, \{2\}, \{7\}, \{8\})$

L4: $L[0]: 2; L[4]: 0; L[6]: 0;$

L5: $H = (\{3\}, \{5\}, \{1\}, \{2\}, \{7\}, \{8\}, \{4, 6\}, \{0\})$

L6: $P = (\{3\}, \{5\}, \{1\}, \{2\}, \{7\}, \{8\}, \{4, 6\}, \{0\})$

L7: $f_p = \{3 \rightarrow 0, 5 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 3, 7 \rightarrow 4, 8 \rightarrow 5, 4 \rightarrow 6, 6 \rightarrow 6, 0 \rightarrow 8\}$

L8: split=TRUE $\rightarrow$ repeat

L3: H=()

L4: L[4]: 0; L[6]: 1;

L5: H=({3}, {5}, {1}, {2}, {7}, {8}, {4}, {6}, {0})

L6: P=({3}, {5}, {1}, {2}, {7}, {8}, {4}, {6}, {0})

L7: $f_p = \{3 \rightarrow 0, 5 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 3, 7 \rightarrow 4, 8 \rightarrow 5, 4 \rightarrow 6, 6 \rightarrow 7, 0 \rightarrow 8\}$

L8: split=TRUE $\rightarrow$ repeat

L3: H=()

... (Repetition of steps L4 to L6 without further refinement)

L7: $f_p = \{3 \rightarrow 0, 5 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 3, 7 \rightarrow 4, 8 \rightarrow 5, 4 \rightarrow 6, 6 \rightarrow 7, 0 \rightarrow 8\}$

L8: split=FALSE $\rightarrow$ stop

Because the final partition contains one vertex in each class, the resulting prenumbering will canonically number the vertices of the graph.

The refinement distributes information concerning local symmetry distortion over the whole structure. Therefore, it allows differentiation of more classes of vertices than by using only local properties.

4.3.3. Refinement of GM-Descriptions

This basic idea was also used in the canonization procedure for GM-descriptions but, because neighborhood implies a binary relation, the graph refinement approach had to be modified.

The following points guided the development of the algorithm:

- Refine center and unit partitions simultaneously.

- Use ordering of unit blocks and attributes to construct primary unit partition.
- Use permutationally minimized tuples of center prenumbers to refine unit partitions.
- Use lists of pairs (unit-prenumber/position-within-tuple) to refine center partitions²⁰).

The following procedure refines the partition defined by f_X for the GM-description Q and modifies f_X accordingly. It assumes that the set of block descriptors, the set of possible attributes, as well as its power set are ordered and uses these orderings to refine partitions of units.

procedure refine(Q, f_X)

Input:

$Q=(X, FG, AS, U)$: GM-description with
domain X ,
block-to-group mapping FG ,
attribute set AS , and
unit-set U .

f_X : prenumbering of the domain X of Q .

²⁰) The part "position-within-tuple" is intended to utilize as much information about "symmetry distortion" as possible for the distinction of centers. An otherwise difficult case would be, for instance, a situation with geminal dimethyl groups.

Declaration:

PX: ordered partition of X;
PU: ordered partition of U;
 f_U : PU-prenumbering of U;
split: flag; /* TRUE if a subset of PX was split, FALSE otherwise */
H: ordered sequence of sets of centers or units;
MX: $|X|$ -vector of lists of pairs;
/* tuple prenumber / index within minimized tuple */
MU: $|U|$ -vector of tuples;
G: permutation group;
p, l: integers;

Initialization:

L1: $PX \leftarrow$ <ordered partition of X corresponding to f_X >;
L2: $PU \leftarrow$ <ordered partition of U according to blocks and attribute sets>;

Splitting center and unit classes:

```
repeat
    H  $\leftarrow$  ();                                /* unit refinement */
    for each set  $c_i \in PU=(c_1, \dots, c_l)$ 
        if  $|c_i| = 1$ 
            H  $\leftarrow H, c_i$ 
        else
            for each unit  $u=(b, A, t) \in c_i$ 
                G  $\leftarrow FG(b)$ ;
L3:         MU[u]  $\leftarrow \min_G(f_X(t))$ ;
            endfor;
L4:         H  $\leftarrow H, <\text{partition of } c_i \text{ refined using lex. comp. of } MU[u]>$ ;
        endif;
    endfor;
L5: PU  $\leftarrow$  H;
L6:  $f_U \leftarrow$  PU-prenumbering;
    H  $\leftarrow$  ();                                /* center refinement */
    split  $\leftarrow$  FALSE;
```

```

for each set  $c_i \in PX = (c_1, \dots, c_l)$ 
  if  $|c_i| = 1$ 
     $H \leftarrow H, c_i$ 
  else
    for each center  $v \in c_i$ 
       $MX[v] \leftarrow ()$ ;
      for each unit  $u = (b, A, t)$  with  $v \in t$ 
         $G \leftarrow FG(b)$ ;
         $p \leftarrow f_U(u)$ ;
        if there is only one  $g \in G$  with  $g(f_X(t)) = \min_G(f_X(t))$ 
           $l \leftarrow \langle \text{index of } v \text{ in } g(t) \rangle^{21}$ ;
        else
           $l \leftarrow 0$ ;
        endif;
         $MX[v] \leftarrow MX[v], (p, l)$ ;
      endfor;
    L7: sort  $MX[v]$ ;
    endfor;
    if all lists  $MX[v]$ ,  $v \in c_i$  are equal
       $H \leftarrow H, c_i$ ;
    else
      L8:  $H \leftarrow H, \langle \text{ordered partition of } c_i \text{ using lex. comp. of } MX[v] \rangle$ ;
      split  $\leftarrow \text{TRUE}$ ;
    endif;
  endfor;
L9:  $PX \leftarrow H$ ;
L10:  $f_X \leftarrow \text{PX-prenumbering}$ ;
      if  $|c_i| = 1$  for all  $c_i \in PX$ 
        split  $\leftarrow \text{FALSE}$ ;

```

21) Index origin 0

```
endif;
L11:  until not split;
      return(fX);
```

This algorithm will now be applied to the pruned structure of Fig. 4-7. The numbers and prenumbers of units are underlined, to distinguish them from center numbers and prenumbers.



Fig. 4-7

Input Parameters:

Q:

Units:

Cross Reference:

<u>0</u> :	IDENT	F	H0	4	0)	<u>4</u>	<u>8</u>	<u>10</u>	
<u>1</u> :	IDENT	C	H1	1	1)	<u>1</u>	<u>5</u>	<u>6</u>	<u>8</u> <u>9</u>
<u>2</u> :	IDENT	C	H2	2	2)	<u>2</u>	<u>5</u>	<u>7</u>	<u>10</u>
<u>3</u> :	IDENT	C	H2	3	3)	<u>3</u>	<u>6</u>	<u>7</u>	<u>10</u>
<u>4</u> :	IDENT	H	H0	0	4)	<u>0</u>	<u>9</u>	<u>10</u>	
<u>5</u> :	BOND	single	r3b	1 2					
<u>6</u> :	BOND	single	r3b	1 3					
<u>7</u> :	BOND	single	r3b	2 3					
<u>8</u> :	BOND	single	chainb	0 1					
<u>9</u> :	BOND	single	chainb	1 4					
<u>10</u> :	ORIENT	plus		0 2 3 4					

f_X: {0→0, 1→0, 2→0, 3→0, 4→0}.

Trace of Algorithm:

L1: $PX = (\{0, 1, 2, 3, 4\})$
 L2: $PU = (\{0\}, \{1\}, \{2, 3\}, \{4\}, \{5, 6, 7\}, \{8, 9\}, \{10\})$
 L3: $MU[2] = (0), MU[3] = (0)$
 L4: $H = (\{0\}, \{1\}, \{2, 3\})$
 L3: $MU[5] = (0, 0), MU[6] = (0, 0), MU[7] = (0, 0)$
 L4: $H = (\{0\}, \{1\}, \{2, 3\}, \{4\}, \{5, 6, 7\})$
 L3: $MU[8] = (0, 0), MU[9] = (0, 0)$
 L4: $H = (\{0\}, \{1\}, \{2, 3\}, \{4\}, \{5, 6, 7\}, \{8, 9\})$
 L5: $PU = (\{0\}, \{1\}, \{2, 3\}, \{4\}, \{5, 6, 7\}, \{8, 9\}, \{10\})$
 L6: $f_U = \{0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 2, 4 \rightarrow 4, 5 \rightarrow 5, 6 \rightarrow 5, 7 \rightarrow 5, 8 \rightarrow 8, 9 \rightarrow 8, 10 \rightarrow 10\}$
 L7: $MX[0] = ((4, 0), (8, 0), (10, 0)),$
 $MX[1] = ((1, 0), (5, 0), (5, 0), (8, 0), (8, 0)),$
 $MX[2] = ((2, 0), (5, 0), (5, 0), (10, 0)),$
 $MX[3] = ((2, 0), (5, 0), (5, 0), (10, 0)),$
 $MX[4] = ((0, 0), (8, 0), (10, 0))$
 L8: $H = (\{4\}, \{1\}, \{2, 3\}, \{0\})$
 L9: $PX = (\{4\}, \{1\}, \{2, 3\}, \{0\})$
 L10: $f_X = \{4 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 2, 0 \rightarrow 4\}$
 L11: $split = TRUE \rightarrow repeat$

 L3: $MU[2] = (2), MU[3] = (2)$
 L4: $H = (\{0\}, \{1\}, \{2, 3\})$
 L3: $MU[5] = (1, 2), MU[6] = (1, 2), MU[7] = (2, 2)$
 L4: $H = (\{0\}, \{1\}, \{2, 3\}, \{5, 6\}, \{7\})$
 L3: $MU[8] = (1, 4), MU[9] = (0, 1)$
 L4: $H = (\{0\}, \{1\}, \{2, 3\}, \{5, 6\}, \{7\}, \{9\}, \{8\})$
 L5: $PU = (\{0\}, \{1\}, \{2, 3\}, \{5, 6\}, \{7\}, \{9\}, \{8\}, \{10\})$
 L6: $f_U = \{0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 2, 4 \rightarrow 4, 5 \rightarrow 5, 6 \rightarrow 5, 7 \rightarrow 7, 9 \rightarrow 8, 8 \rightarrow 9, 10 \rightarrow 10\}$

At the next break point, the importance of the second component of the pairs in the MX-lists is demonstrated. The tuple of the unit number 10

distinguishes the centers 2 and 3. The tuple - with the corresponding prenumbering pointed to by arrows - is: (0→4, 2→2, 3→2, 4→0). The permutationally (by application of elements of the A_4) minimized tuple is (4→0, 3→2, 2→2, 0→4) and this minimum is unique because there is no permutation $\in A_4$, which exchanges only two positions in the tuple. Therefore, the index of the center 3 in the tuple 10 is 1 and that of center 2 is 2.

```
L7:      MX[2]=((2, 0), (5, 1), (7, 0), (10, 2))
          MX[3]=((2, 0), (5, 1), (7, 0), (10, 1))
L8:      H=({4}, {1}, {3}, {2})
L9:      PX=({4}, {1}, {3}, {2}, {0})
L10:     f_X={4→0, 1→1, 3→2, 2→3, 0→4}
L11:     split=FALSE /* partition completely refined */ → return(f_X)
```

The renumbered GM-description:

Units:

IDENT	F	H0	0
IDENT	C	H1	1
IDENT	C	H2	2
IDENT	C	H2	3
IDENT	H	H0	4
BOND	single	r3b	1 2
BOND	single	r3b	1 3
BOND	single	r3b	2 3
BOND	single	chainb	0 1
BOND	single	chainb	1 4
ORIENT	plus		0 2 3 4



Fig. 4-8

Fig. 4-8 shows the renumbered molecule of Fig. 4-7.

4.3.4. The Automorphism Group of Symmetrical Molecules

In the previous examples illustrating refinement algorithms, the procedure (which started with all nodes or centers in one class) stopped with partitions containing only singleton sets. Consequently, they fully define a canonical (input numbering independent) numbering, which can then be used to generate a unique code. This is, of course, only possible, if all the atoms are different, i.e., if the molecule has no (proper) symmetry element mapping one atom onto another.

The symmetry of a molecule is reflected in the so-called automorphisms of its GM-description. These are renumberings which lead to congruent GM-descriptions.

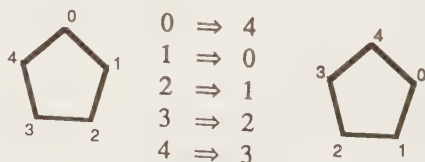


Fig. 4-9

Fig. 4-9 shows the effect of such a renumbering on cyclopentane. The automorphism corresponds to a rotation around the center of the pentagon. This renumbering can also be described as the permutation (0 1 2 3 4)

written in cycle notation, which is a cyclic interchange of the numbers 0, 1, 2, 3, and 4. The fact that the rotation around the center is a symmetry operation of this structure shows up in its GM-description as follows:

The initial GM-description is:

IDENT	C H2	0
IDENT	C H2	1
IDENT	C H2	2
IDENT	C H2	3
IDENT	C H2	4
BOND	single r5b	0 1
BOND	single r5b	0 4
BOND	single r5b	1 2
BOND	single r5b	2 3
BOND	single r5b	3 4

Applying $f=\{0\rightarrow 4, 1\rightarrow 0, 2\rightarrow 1, 3\rightarrow 2, 4\rightarrow 3\}$:

IDENT	C H2	4
IDENT	C H2	0
IDENT	C H2	1
IDENT	C H2	2
IDENT	C H2	3
BOND	single r5b	4 0
BOND	single r5b	4 3
BOND	single r5b	0 1
BOND	single r5b	1 2
BOND	single r5b	2 3

Permutationally minimizing tuples:

IDENT	C H2	4
IDENT	C H2	0
IDENT	C H2	1
IDENT	C H2	2
IDENT	C H2	3

```

BOND  single r5b  0 4
BOND  single r5b  3 4
BOND  single r5b  0 1
BOND  single r5b  1 2
BOND  single r5b  2 3

```

Sorting tuple set to obtain a standardized code for the list:

```

IDENT  C H2      0
IDENT  C H2      1
IDENT  C H2      2
IDENT  C H2      3
IDENT  C H2      4
BOND   single r5b 0 1
BOND   single r5b 0 4
BOND   single r5b 1 2
BOND   single r5b 2 3
BOND   single r5b 3 4

```

The symmetry group of a molecule is the group of all those mappings. In this case it is:

$$G = \{ (0)(1)(2)(3)(4), (0\ 1\ 2\ 3\ 4), (0\ 2\ 4\ 1\ 3), (0\ 3\ 1\ 4\ 2), \\ (0\ 4\ 3\ 2\ 1), (0)(1\ 4)(2\ 3), (1)(0\ 2)(3\ 4), (2)(0\ 4)(1\ 3), \\ (3)(0\ 1)(2\ 4), (4)(0\ 3)(1\ 2) \}.$$

The presence of symmetry also causes problems in canonical numbering:

If Q is a GM-description with the automorphism group G , i.e., $g(Q) \equiv Q$ with $g \in G$, and if f is a renumbering of Q which derives the canonical GM-description, then $f(Q) \equiv f(g(Q))$, i.e. $f \circ g$ also yields a canonical GM-description²²).

On the other hand, if f and g are two renumberings with $f(Q) \equiv g(Q)$, then $g^{-1}(f(Q)) \equiv Q$, i.e. $g^{-1} \circ f$ is an automorphism of Q .

For cyclopentane this means that there are not only one but ten canonical numberings which result in congruent GM-descriptions.

This problem becomes even more severe if more symmetry elements are present. Adding a *t*-butyl-group as substituent to a molecule, for instance, usually increases the size of the automorphism group three-fold²³).

Naming every element is not an efficient way to define a permutation group. Mathematicians, therefore, developed the concept of generating sets.

Definition:

Let G be a group with multiplication " $\circ$ ". A subset K of G is called a generating set for G if every $g \in G$ can be expressed as $g = k_1 \circ k_2 \circ \dots \circ k_l$ with $k_1, \dots, k_l \in K$. The members of K are called generators.

Generating sets can be quite small even for large groups. For instance, the S_{10} , which contains $10! = 3,628,800$ elements can be generated by only two permutations, namely $(0\ 1)(2)(3)(4)(5)(6)(7)(8)(9)$ and $(0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)$ [44].

22) " $\circ$ " means function composition.

23) If a molecule were not hydrogen pruned, even an added methyl-group would have that effect.

4.3.5. Straightforward Canonization Procedure

A canonization procedure which computes a renumbering of a GM-description and simultaneously finds its automorphism group can be built as follows:

- Start with the trivial partition of the centers, i.e. all centers in one class. This means that the prenumbers of all centers are 0.
- Call a procedure that refines the partition.
- If ambiguities remain, try all possible selections of one element of the lowest class to get the smallest number, and recursively call the same procedure.
- If identical codes are encountered, compute the corresponding element of the automorphism group.
- On completion, return one canonical renumbering and the elements of the automorphism group.

The following pseudocode function shows how this sketch is cast into an algorithm. It uses a subfunction $\text{compare}(Q, f_X, g_X)$, which compares the codes of Q generated by the two renumberings f_X and g_X , and returns 1, 0, or (-1) if $f_X(Q)$ is greater than, equal to, or less than $g_X(Q)$, respectively.

function canonrec($Q, f_X, \text{Aut}, f_X^0$)

Input:

$Q = (X, FG, AS, U)$: GM-description with
domain X ,
block-to-group mapping FG ,

attribute set AS, and
unit-set U.

f_X : prenumbering of X^{24}).

f_X^0 : candidate for canonical renumbering from previously
examined possibilities²⁵).

Output:

Aut: Subgroup of automorphisms of Q for which $f_X(g(x))=f_X(x)$
for all $g \in \text{Aut}$ and all $x \in X$.

f_X^0 : candidate for canonical renumbering after that branch of
the tree of possibilities has been searched.

Return:

- 1 if the best new code was $> f_X^0(Q)$,
- 0 if the best new code was $= f_X^0(Q)$, and
- 1 if a better code was found.

Declarations:

G: set of permutations of X .

h_X : prenumbering of X .

PX: partition of X .

TBT: subset of X , to be tested for being assigned a certain
number

rcomp,

comp: integers, bearing values (-1), 0, or 1.

²⁴) When the canonization starts at the top level of recursion, f_X is set to $\{0 \rightarrow 0, 1 \rightarrow 0, \dots, |X| - 1 \rightarrow 0\}$.

²⁵) If no previous search has been done, f_X^0 is set to $\{0 \rightarrow |X|, 1 \rightarrow |X|, \dots, |X| - 1 \rightarrow |X|\}$ to ensure that $f_X^0(Q)$ is greater than any candidate code $f_X(Q)$.

Canonization:

```

    refine( $Q, f_X$ );
     $PX \leftarrow$  <partition of  $X$  corresponding to  $f_X$ >;
    if  $|c_i| = 1$  for all  $c_i \in PX$     /* Numbering completely refined */
        comp  $\leftarrow$  compare( $Q, f_X, f_X^o$ );
        if comp = (-1) then
             $f_X^o \leftarrow f_X$ ;
            Aut  $\leftarrow \{e\}$ ;
        elseif comp = 0 then
            Aut  $\leftarrow \{f_X^{-1} \circ f_X^o\}$ ;
        endif;
        return(comp);
    else
        Aut  $\leftarrow \{\}$ ; rcomp  $\leftarrow 1$ ;
        TBT  $\leftarrow$  <class  $c_i \in PX$  with the lowest  $i$  for which  $|c_i| > 1$ >;
        for all  $y \in TBT$  do          /* Try all possibilities for  $y$  */
             $h_X \leftarrow f_X$ ;
            for all  $z \in c_i$  do
                if  $z \neq y$  then
                     $h_X(z) \leftarrow f_X(y) + 1$ ;
                endif;
            endfor;
            comp  $\leftarrow$  canonrec( $Q, h_X, G, f_X^o$ );
            if comp = (-1) then
                Aut  $\leftarrow G$ ;
            else if comp = 0 then
                Aut  $\leftarrow$  Aut  $\cup G$ ;
            endif;
            rcomp  $\leftarrow$  min(rcomp, comp);
        endfor;
        return(rcomp);
    endif;

```

Cyclobutane (Fig. 4-10) was selected to illustrate the algorithm.



Fig. 4-10

The corresponding GM-description Q is:

IDENT	C H2	0
IDENT	C H2	1
IDENT	C H2	2
IDENT	C H2	3
BOND	single r4b	0 1
BOND	single r4b	0 3
BOND	single r4b	1 2
BOND	single r4b	2 3

A trace of the algorithm would be rather long because there are eight equivalent renumberings for the molecule resulting in the same canonical code. Therefore, Fig. 4-11 shows the search tree of prenumberings g_X being traversed during execution of the algorithm.

The edges of this flow graph represent the calling relationships. The input numbering is shown on the top, and the graph below it has nodes, which are labelled with the molecules numbered with the appropriate prenumberings. Lower case letters are used as additional node labels.

The tree is traversed in a depth first fashion, i.e. $a \rightarrow b \rightarrow f \rightarrow b \rightarrow g \rightarrow b \rightarrow a \rightarrow c \rightarrow h \rightarrow c \rightarrow i \rightarrow c \rightarrow a \rightarrow d \rightarrow j \rightarrow d \rightarrow k \rightarrow d \rightarrow a \rightarrow e \rightarrow l \rightarrow e \rightarrow m \rightarrow e \rightarrow a \rightarrow \text{stop}$.

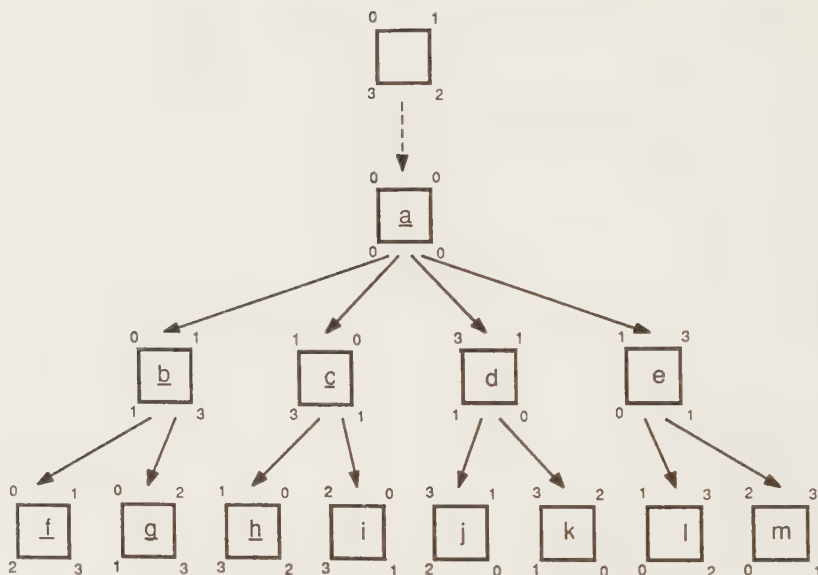


Fig. 4-11

The canonical renumbering returned as f_X^0 is $\{0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 3, 3 \rightarrow 2\}$ with the corresponding canonical code:

IDENT	C H2	0
IDENT	C H2	1
IDENT	C H2	2
IDENT	C H2	3
BOND	single r4b	0 1
BOND	single r4b	0 3
BOND	single r4b	1 2
BOND	single r4b	2 3

The automorphism group Aut is $\{f^{-1} \circ f, g^{-1} \circ f, \dots, m^{-1} \circ f\} = \{(0)(1)(2)(3), (0)(1\ 3)(2), \dots, (0\ 3\ 2\ 1)\}^{26}$.

4.3.6. Canonization Using Generators

It was mentioned before that a group, (such as an automorphism group) needs not be described by enumerating all of its elements, but that a generating set suffices for some applications.

In the example of Fig. 4-11, the more refined algorithm developed in this dissertation can find the renumbering to the canonical code, while examining only the numberings labelled with underscored letters and computing only two generators of the automorphism group of cyclobutane. The following shows how they are found:

After traversing the path $\underline{a} \rightarrow \underline{b} \rightarrow \underline{f}$, a renumbering is found. It happens to be the canonical one, but that is not yet known, because there are still branches of the search tree to be examined. Backtracking to $\underline{b}$ and then examining $\underline{g}$ yields $(0)(1\ 3)(2)$ as an element of the automorphism group. Therefore, when processing continues at the $\underline{a}$ -node, it is known that atom 1 is equivalent to atom 3. This means: If atom 1 would have been tested for receiving number 0, and this would result in the same code, it could be concluded that testing atom 3 would yield the same result.

Descending subsequently to node $\underline{h}$ establishes $(0\ 1)(2\ 3)$ as a second generator. At root node $\underline{a}$ of the search tree, we now have the equivalences $1 \hat{=} 3$, and $0 \hat{=} 1$, $2 \hat{=} 3$, which means that all atoms are equivalent. So the same code that was obtained by setting the new number of atom 0 to 0 would also be obtained from the three other assignments.

²⁶⁾ The product $m^{-1} \circ f$ for instance is computed as follows: $f = \{0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 3, 3 \rightarrow 2\}$, $m = \{0 \rightarrow 1, 1 \rightarrow 3, 2 \rightarrow 2, 3 \rightarrow 0\}$, $m^{-1} = \{0 \rightarrow 3, 1 \rightarrow 0, 2 \rightarrow 2, 3 \rightarrow 1\}$, so $m^{-1} \circ f = \{0 \rightarrow 0 \rightarrow 3, 1 \rightarrow 1 \rightarrow 0, 2 \rightarrow 3 \rightarrow 1, 3 \rightarrow 2 \rightarrow 2\}$.

Therefore, only the underscored nodes of the tree need to be examined and the search can be ended with this result:

$f_X^0 = \{0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 3, 3 \rightarrow 2\}$, and
 $\text{Aut}(Q)$ is generated by $\{(0)(1\ 3)(2), (0\ 1)(2\ 3)\}$.

The ideas, which emerged from this example, have been used to improve the algorithm described above (changes and additions are marked by '|' at the beginning of the lines.):

function canonrec($Q, f_X, \text{Aut}, f_X^0$)

Input:

$Q = (X, FG, AS, U)$: GM-description with
 domain X ,
 block-to-group mapping FG ,
 attribute set AS
 unit-set U , and

f_X : prenumbering of X .

f_X^0 : candidate for canonical renumbering from previously
 examined possibilities.

Output:

| Aut : Generating set for the subgroup of automorphisms of Q
 | for which $f_X(g(x)) = f_X(x)$ for all $g \in \text{Aut}$ and all $x \in X$.

f_X^0 : candidate for canonical renumbering after that branch of
 the tree of possibilities has been searched.

Return:

1 if the best new code was $> f_X^0(Q)$,
 0 if the best new code was $= f_X^0(Q)$, and
 -1 if a better code was found.

Declarations:

G: set of permutations of X.
 h_X : prenumbering of X.
 PX: partition of X.
 TBT: subset of X, to be tested for being assigned a certain number.
 | Tested: subset of X; contains all tested atoms and their known symmetry equivalents.
 rcomp,
 comp: integers, bearing values (-1), 0, or 1.
 | branched: Boolean; set TRUE if searching for an automorphism.

Canonization:

```

refine(Q,  $f_X$ );
PX  $\leftarrow$  <partition of X corresponding to  $f_X$ >;
if  $|c_i| = 1$  for all  $c_i \in PX$  /* Numbering completely refined */
    comp  $\leftarrow$  compare(Q,  $f_X$ ,  $f_X^o$ );
    if comp = (-1) then
         $f_X^o \leftarrow f_X$ ;
        Aut  $\leftarrow \{\}$ ;
    elseif comp = 0 then
        Aut  $\leftarrow \{f_X^{-1} \circ f_X^o\}$ ;
    endif;
    return(comp);
else
    | rcomp  $\leftarrow$  1;
    | branched  $\leftarrow$  FALSE;
    TBT  $\leftarrow$  <class  $c_i \in PX$  with the lowest i for which  $|c_i| > 1$ >;
    for all y  $\in$  TBT do /* Try all possibilities for y */
         $h_X \leftarrow f_X$ ;
        for all z  $\in$   $c_i$  do
            if z  $\neq$  y then
                 $h_X(z) \leftarrow f_X(y) + 1$ ;

```

```

        endif;
    endfor;
    comp ← canonrec(Q, hX, G, fXo);
|   Tested ← Tested ∪ {y};
    if comp = (-1) then
        Aut ← G;
|       branched ← TRUE;
|   elseif comp = 0 then
        Aut ← Aut ∪ G;
|       if not branched then
|           return(comp);
        endif;
|   Tested ← Tested ∪ Aut-equivalent(Tested);
|   TBT ← TBT \ Tested;
    rcomp ← min(rcomp, comp);
endfor;
return(rcomp);
endif;

```

If ringsize information is included in the GM-descriptions, examples wherein the refined partitions are not equal to the automorphism partitions rarely occur. To illustrate the behavior of the algorithm in case of incomplete refinement, the graph²⁷⁾ on the top of Fig. 4-12 was, therefore, canonized neglecting ring size information. Its GM-description Q is:

```

BOND single  0 1
BOND single  0 3
BOND single  0 7
BOND single  1 2

```

²⁷⁾ A graph, instead of a chemical structure, was chosen to focus on the main points, so that non-BOND units need not be considered.

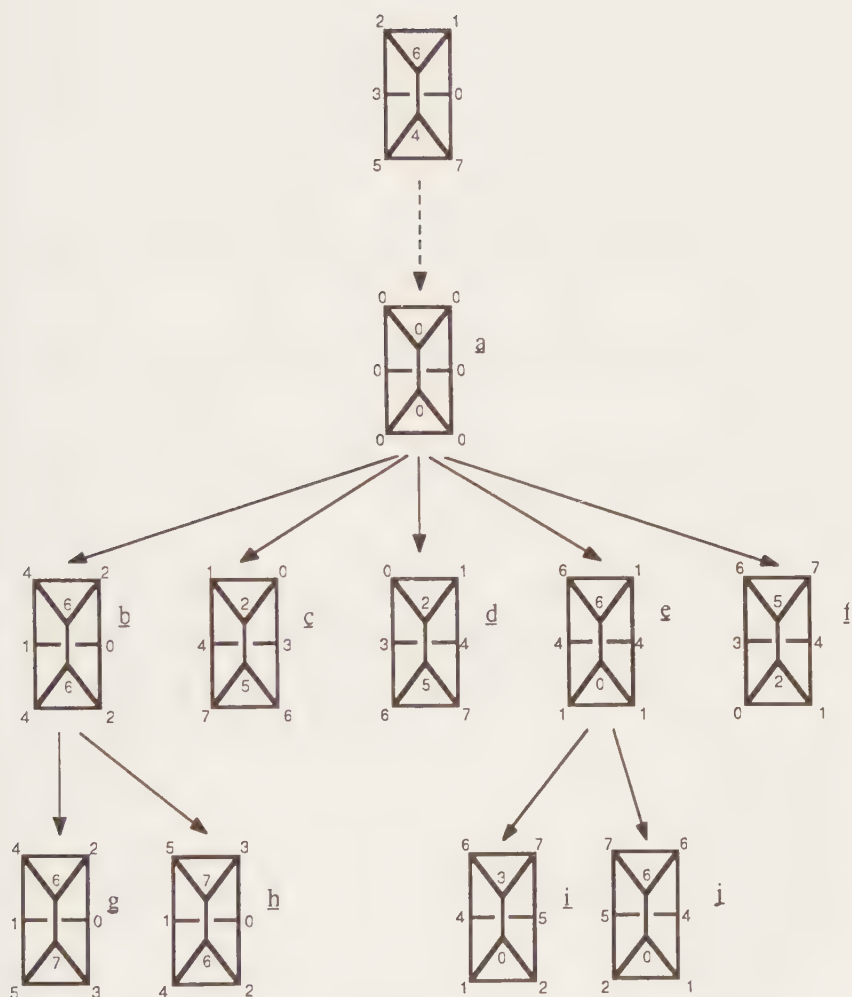


Fig. 4-12

```
BOND single 1 6
BOND single 2 3
BOND single 2 6
BOND single 3 5
BOND single 4 5
BOND single 4 6
BOND single 4 7
BOND single 5 7
```

The figure shows the search tree of prenumberings examined during canonization.

At the top level a all nodes of the graph are considered equivalent by the refinement procedure, so all of them are candidates for receiving number 0.

The first node tested is node 0 and the resulting prenumbering is shown in b. Assigning 2 to node 1 or node 7 results in the renumberings g and h, respectively, which both define the same code:

```
BOND single 0 1
BOND single 0 2
BOND single 0 3
BOND single 1 4
BOND single 1 5
BOND single 2 4
BOND single 2 6
BOND single 3 5
BOND single 3 7
BOND single 4 6
BOND single 5 7
BOND single 6 7
```

Back at node a, we have:

$$\text{Aut} = \{(0)(1\ 7)(2\ 5)(3)(4\ 6)\},$$

$$f_X^0 = \{0 \rightarrow 0, 1 \rightarrow 2, 2 \rightarrow 4, 3 \rightarrow 1, 4 \rightarrow 7, 5 \rightarrow 5, 6 \rightarrow 6, 7 \rightarrow 3\}, \text{ and}$$

$$\text{Tested} = \{0\}.$$

Then node 1 is assigned number 0, refinement (at c) ends with the renumbering $f_X = \{0 \rightarrow 3, 1 \rightarrow 0, 2 \rightarrow 1, 3 \rightarrow 4, 4 \rightarrow 5, 5 \rightarrow 7, 6 \rightarrow 2, 7 \rightarrow 6\}$, and the new code is:

```

BOND single  0 1
BOND single  0 2
BOND single  0 3
BOND single  1 2
BOND single  1 4
BOND single  2 5
BOND single  3 4
BOND single  3 6
BOND single  4 7
BOND single  5 6
BOND single  5 7
BOND single  6 7

```

This code is lexicographically smaller than the previous one (comp=(-1)).

Thus, at the a level, Aut is emptied and the variables contain:

$$\text{Aut} = \{\},$$

$$f_X^0 = \{0 \rightarrow 3, 1 \rightarrow 0, 2 \rightarrow 1, 3 \rightarrow 4, 4 \rightarrow 5, 5 \rightarrow 7, 6 \rightarrow 2, 7 \rightarrow 6\}, \text{ and}$$

$$\text{Tested} = \{0, 1\}.$$

Examining the assignment $2 \rightarrow 0$ discloses $(0\ 3)(1\ 2)(4)(5\ 7)(6)$ as a new generator of the automorphism group of Q , which means that $0 \triangleq 3$, $1 \triangleq 2$, and $5 \triangleq 7$; because the node 0 has already been tried and $0 \triangleq 3$, node 3

needs not to be analyzed and can be directly included into the set Tested. The next possible assignment is $4 \rightarrow 0$. It results in the two renumberings $\underline{i}$ and $\underline{j}$ both generating the same code, which is lexicographically larger than $f_X^0(Q)$:

BOND single	0 1
BOND single	0 2
BOND single	0 3
BOND single	1 2
BOND single	1 4
BOND single	2 5
BOND single	3 6
BOND single	3 7
BOND single	4 5
BOND single	4 6
BOND single	5 7
BOND single	6 7

After this test the variable Tested contains the set $\{0, 1, 2, 3, 4\}$ and node 5 is assigned number 0. The refinement produces a code equivalent to f_X^0 and the generator $(0\ 3)(1\ 5)(2\ 7)(4\ 6)$ is found. From both generators now contained in Aut, the nodes can be divided into three sets of equivalent nodes, namely $\{0, 3\}$, $\{1, 2, 5, 7\}$, and $\{4, 6\}$. Because the nodes 6 and 7 are equivalent to nodes already tested, the algorithm stops with this result:

Aut = $\{(0\ 3)(1\ 2)(4)(5\ 7)(6), (0\ 3)(1\ 5)(2\ 7)(4\ 6)\}$ and
 $f_X^0 = \{0 \rightarrow 3, 1 \rightarrow 0, 2 \rightarrow 1, 3 \rightarrow 4, 4 \rightarrow 5, 5 \rightarrow 7, 6 \rightarrow 2, 7 \rightarrow 6\}$.

In this (artificial) example, seven renumberings had to be tested, instead of twelve for the "straightforward" canonizer. With ring information included, only the renumberings $\underline{c}$, $\underline{d}$, and $\underline{f}$ have to be examined.

Usually, an additional symmetry element (at least) doubles the number of equivalent renumberings, but it only adds one or two generators to the generating set accumulated by the canonizer described above. In the chapter on performance, several classes of graphs are canonized to show the time requirements of an implementation of this algorithm in the C-language.

4.4. Substructure Testing

The notion 'generalized M-description' was developed to be able to describe molecules as well as substructures. The subject of this section is an algorithm to test whether one GM-description Q_1 is a sub-GM-description of Q_2 (written $Q_1 \leq Q_2$). This procedure will be called sub-GM-testing²⁸).

The main ideas of this algorithm are derived from algorithms used in subgraph isomorphism testing [47]. Another related class of algorithms is concerned with constraint satisfaction [48,49], which can be also regarded as a generalization of the subgraph isomorphism testing. The following subsection, therefore, briefly presents the ideas developed in that work.

4.4.1. Subgraph Isomorphism Testing

Chemical problems - as well as image analysis and pattern recognition - have triggered most of the investigations on subgraph isomorphism. The terminology used is rather non-uniform: For instance, atom-by-atom [50] or node-by-node [51] matching are similar to resolution [53], and set reduction [52] corresponds to relaxation [53,54].

²⁸) Because GM-description are generalizations of graphs, sub-GM-testing is NP-complete [46]. This property makes it very unlikely that there will be a general algorithm for sub-GM-testing that has a running time bounded by a polynomial in the size of the input descriptions.

Aside from these terminological differences, the procedures for subgraph isomorphism testing discussed in the literature use the same general ideas:

- Classification of vertices
- Construction of a mapping of the vertices of the subgraph on sets of vertices of the supergraph using compatibility of vertex classes
- Refinement of those mappings (i.e., reduction of the size of the image sets) by applying the neighborhood relationship (achieving local consistency).
- If one of the image sets becomes empty, no subgraph isomorphism is possible.
- If a subgraph vertex remains that maps to a set containing more than one supergraph vertex, all possible one to one mappings of that vertex to the vertices in the set are tried until either all possibilities lead to a contradiction, or a subgraph isomorphism is found.

The published algorithms differ:

- In the properties used for primary classification.
- In the implementation of the one-to-many mappings.
- In the effort spent during refinement.
- In the procedure applied to split the image set of a vertex.

29) Sussenguth calls the vertices of the graph of a molecule nodes and this nomenclature is adopted in the description of his algorithm.

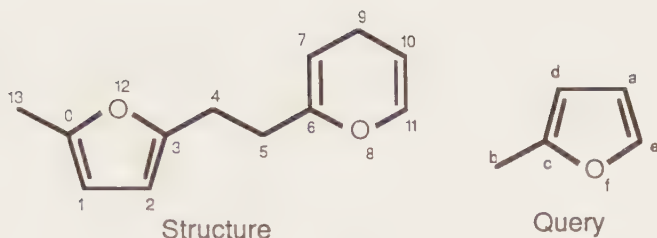


Fig. 4-13

The classical algorithm of Sussenguth²⁹⁾ will be described here as an example. Fig. 4-13 shows a molecule labelled with numbers and a query labelled with letters. The algorithm is to find a mapping $f: X = \{a, b, c, d, e, f\} \rightarrow Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}$, such that, for all $x \in X$, the atom type (or node value) of x is equal to the atom type of $f(x)$, and that, for all bonds (x, y) , $x, y \in X$, $f(x)$ and $f(y)$ are bonded with the same bond order as x and y .

Sussenguth used the following properties to classify the nodes (i.e. the atoms) and restrict the initial mapping of query nodes onto sets of structure nodes:

Node value (atom type):

The atom types are used as node values to build classes, because a query atom of a certain atom type can only be mapped on atoms of the same type in the structure.

Branch value (bond type or bond order):

All query atoms connected to some other atom by a bond of order b can only be mapped to structure atoms also bearing a bond of order b ³⁰⁾.

Degree (ligancy):

The number of bonds of an atom in a structure or a query is called its degree. The degree of a query atom must be less than, or equal to any structure atom it is to be mapped on.

Order:

The order of an atom is the size of the smallest ring containing that atom. The order of chain atoms is ∞ by definition. The order of a query atom must be greater than, or equal to the order of a corresponding structure atom.

Connectivity:

If an atom a of the query is mapped on an atom b of the structure, then the atoms connected to a must be mapped on atoms connected to b.

The first four properties and the set correspondences following from them for Fig. 4-13 are summarized in Table IV-1. The next step in Sussenguth's algorithm is the partitioning:

If it is known that, for all i in some set of indices, a query node x corresponds to a node in B_i , then it can be concluded that x must correspond to a node in the intersection of all B_i .

30) This is the only place in Sussenguth's algorithm, where bond orders are utilized. Ming and Tauber [55] pointed out that this means that $=CH-CH=$ could not be differentiated from $-CH=CH-$, and that the algorithm fails to produce the desired result. This is usually not a serious problem, because of the nature of the queries to be expected in chemistry. The two atoms are sp^2 -atoms in either case and the exact location of the double bonds might be unimportant because of possible delocalization.

Table IV-1

property of query nodes	query subset	corresponding structure subset	pair number
atom type			
C	{a, b, c, d, e}	{0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 13}	1
O	{f}	{8, 12}	2
branch value			
single	{a, b, ..., f}	{0, ..., 13}	3
double	{a, c, d, e}	{0, 1, 2, 3, 6, 7, 10, 11}	4 4
degree			
1	{b}	{0, ..., 13}	5
2	{a, d, e, f}	{0, ..., 12}	6
3	{c}	{0, 3, 6}	7
order			
5	{a, c, d, e, f}	{0, 1, 2, 3, 12}	8
6	{}	{0, 1, 3, 6, ..., 12}	9
∞	{b}	{0, ..., 13}	10

Furthermore, if set A corresponds to a set B and $|A| = |B|$, then A' (the complement of A in the set of query atoms) corresponds to B' (the complement of B in the set of structure atoms)³¹). Table IV-2 shows the application of this procedure. Only productive operations are denoted and nodes corresponding to the same set of structure atoms are joined in one set of query atoms.

³¹) This fact is normally used only if $|A|=1$, i.e. if an assignment of one query atom has been established.

Table IV-2

intersection	query set	structure set	pair number
1n4n6n8	{a, d, e}	{0, 1, 2, 3}	11
1n4n7n8	{c}	{0, 3}	12
2n6n8	{f}	{12}	13

After partitioning the neighboring relationship is exploited. For instance, it can be concluded that the neighbors of f, namely c and e, correspond to the neighbors of 12, being 0 and 3. The additional correspondences resulting from these relationships are shown in Table IV-3.

Table IV-3

query set	structure set	query neighbors	structure neighbors
{a, d, e}	{0, 1, 2, 3}	{a, c, d, e, f}	{0, 1, 2, 3, 12, 13}
{c}	{0, 3}	{b, d, f}	{1, 2, 4, 12, 13}
{f}	{12}	{c, e}	{0, 3}

Table VI-4 shows the resulting mappings:

Table VI-4

query set	structure set
{a}	{0, 1, 2, 3}
{b}	{1, 2, 4, 13}
{c, e}	{0, 3}
{d}	{1, 2}
{f}	{12}

Because c and e map on $\{0, 3\}$ the node a, which is in the complement of $\{c, e\}$, cannot map on either of these, so 0 and 3 can be excluded from the images of a. The same argument, applied to the images of a and b, then restricts the image set of b to $\{4, 13\}$. The final stage of the set reduction is shown in Table VI-5.

Table VI-5

query set	structure set
$\{a, d\}$	$\{1, 2\}$
$\{b\}$	$\{4, 13\}$
$\{c, e\}$	$\{0, 3\}$
$\{f\}$	$\{12\}$

The only mapping uniquely defined at this stage is $f \rightarrow 12$. This is due to the fact that there are two mappings of the query nodes on the structure nodes which fulfill the conditions stated above. To resolve this ambiguity an image set of one node is selected, and each of the possible node to node mappings is recursively analyzed further. If all possible trial assignments fail to find a compatible mapping, no subgraph isomorphism can be found, otherwise, in at least one case, a compatible one-to-one mapping of subgraph nodes on graph nodes is computed.

Applying this procedure to the example, we have to test the mappings $a \rightarrow 1$ and $a \rightarrow 2$. Repeatedly using the neighborhood relationship and the partitioning procedure, we end up with the two subgraph isomorphisms $\{a \rightarrow 1, b \rightarrow 4, c \rightarrow 3, d \rightarrow 2, e \rightarrow 0, f \rightarrow 12\}$ and $\{a \rightarrow 2, b \rightarrow 13, c \rightarrow 0, d \rightarrow 1, e \rightarrow 3, f \rightarrow 12\}$.

The alternating application of relaxation (to obtain local consistency [48]) and case analysis is typical of many of the algorithms, which solve the problems cited above. A general discussion of this type of algorithm can be found in [49].

4.4.2. Relaxation for Sub-GM-Testing

According to the definition (see Chapter 3), $Q_q \leq Q_s$ can be established by constructing two functions f_X and f_U , f_X mapping the domain of the GM-description Q_q on the domain of Q_s , and f_U mapping the unit-set of Q_q on the unit set of Q_s .

The algorithm used in GM-SEARCH constructs f_X and f_U (or disproves their existence) simultaneously by a relaxation/case-analysis procedure of the type described above for subgraph isomorphism testing. The case-analysis is very similar to what is already in common use, so that it will be only sketched. The main computation is performed in the relaxation process, which will be described in the following. Rather than giving a detailed - and complex - pseudocode formulation, I will show which local consistency constraints are exploited.

The problem of Fig. 4-14 is used to illustrate the various steps of the procedure.

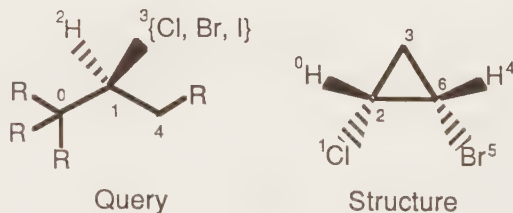


Fig. 4-14

The GM-descriptions for the query (Q_q) and the structure (Q_s) are:

Q_q :

Units:

<u>0</u> :	IDENT	Cl Br I	H0	3
<u>1</u> :	IDENT	C	H0 H1 H2 H3	0
<u>2</u> :	IDENT	C	H1	1

<u>3</u> :	IDENT	C	H2 H3	4
<u>4</u> :	IDENT	H	H0	2
<u>5</u> :	BOND	single	r3b r4b r5b r6b r7b r8b rhb chainb	0 1
<u>6</u> :	BOND	single	r3b r4b r5b r6b r7b r8b rhb chainb	1 2
<u>7</u> :	BOND	single	r3b r4b r5b r6b r7b r8b rhb chainb	1 3
<u>8</u> :	BOND	single	r3b r4b r5b r6b r7b r8b rhb chainb	1 4
<u>9</u> :	ORIENT	plus		0 2 3 4

Cross Reference:

0:	<u>1</u> <u>5</u> <u>9</u>
1:	<u>2</u> <u>5</u> <u>6</u> <u>7</u> <u>8</u>
2:	<u>4</u> <u>6</u> <u>9</u>
3:	<u>0</u> <u>7</u> <u>9</u>
4:	<u>3</u> <u>8</u> <u>9</u>

Q_s:

Units:

<u>0</u> :	IDENT	Br H0	5
<u>1</u> :	IDENT	Cl H0	1
<u>2</u> :	IDENT	C H1	2
<u>3</u> :	IDENT	C H1	6
<u>4</u> :	IDENT	C H2	3
<u>5</u> :	IDENT	H H0	0
<u>6</u> :	IDENT	H H0	4
<u>7</u> :	BOND	single r3b	2 3
<u>8</u> :	BOND	single r3b	2 6
<u>9</u> :	BOND	single r3b	3 6
<u>10</u> :	BOND	single chainb	0 2
<u>11</u> :	BOND	single chainb	1 2
<u>12</u> :	BOND	single chainb	4 6
<u>13</u> :	BOND	single chainb	5 6
<u>14</u> :	ORIENT	plus	0 1 3 6
<u>15</u> :	ORIENT	plus	2 3 4 5

Cross Reference:

0:	<u>5</u> <u>10</u> <u>14</u>
1:	<u>1</u> <u>11</u> <u>14</u>
2:	<u>2</u> <u>7</u> <u>8</u> <u>10</u> <u>11</u> <u>15</u>
3:	<u>4</u> <u>7</u> <u>9</u> <u>14</u> <u>15</u>
4:	<u>6</u> <u>12</u> <u>15</u>
5:	<u>0</u> <u>13</u> <u>15</u>
6:	<u>3</u> <u>8</u> <u>9</u> <u>12</u> <u>13</u> <u>14</u>

In the following $u_q = (b_q, A_q, t_q)$ will denote some element of the query's unit-set with b_q , A_q , and t_q being the corresponding block descriptor, attribute set, and tuple, respectively. Analogously, $u_s = (b_s, A_s, t_s)$ denotes an element of the structure's unit-set. x_q represents some query center and x_s a structure center. The numbers of individual centers and units are indexed with q or s if they belong to the query or the structure, respectively. Sets of centers or units will be indexed as a whole.

During the construction of f_X and f_U , they are approximated by two functions F_X and F_U . $F_X(x)$ is the set of possible image centers of x and $F_U(u)$ is the set of possible image units of u under the sub-GM-isomorphism to be computed. The image sets are represented as bit strings, which can be manipulated using the bit operation primitives of the C-language.

In the subsequent figures, F_X and F_U are represented as $|X_q|$ by $|X_s|$ and $|U_q|$ by $|U_s|$ character matrices, respectively. A '1' in the i -th row and j -th column means a possible mapping of the query center (or unit) number i on the structure center (or unit) number j , whereas '.' says that there is no compatible mapping.

4.4.2.1. Initialization

The first step in the 'matching' procedure computes initial functions F_X and F_U .

$F_U(u_q)$ is initialized to the set of all structure units u_s with $b_s = b_q$ and $A_s \subseteq A_q^{32}$. Fig. 4-15 shows the resulting mapping for the example of Fig. 4-14.

Let $nu(b, x) = |\{u = (b, A, t) \in U \mid x \in t\}|$ be the number of units with block descriptor b containing some center x , and let $nr(x)$ be the number of units $u = (b, A, t)$ where $b = \text{BOND}$, chainb is not $\in A$, and $x \in t$. F_X is then initialized as:

$F_U:$

		-	-	-	-		-	-	-	-		-	-	-	-		
-		1	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.
		.	.	1	1	1	.	.	.	.	.	.	.	.	.	.	.
		.	.	1	1	.	.	.	.	.	.	.	.	.	.	.	.
		.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.
		.	.	.	.	.	1	1	.	.	.	.	.	.	.	.	.
-		.	.	.	.	.	.	1	1	1	1	1	1	1	.	.	.
		.	.	.	.	.	.	1	1	1	1	1	1	1	.	.	.
		.	.	.	.	.	.	1	1	1	1	1	1	1	.	.	.
		.	.	.	.	.	.	1	1	1	1	1	1	1	.	.	.
		.	.	.	.	.	.	.	.	.	.	.	.	.	1	1	.

Fig. 4-15

$$F_X(x_q) = \{x_s \mid \text{nu}(b, x_s) \geq \text{nu}(b, x_q) \text{ for all } b \in \text{BD}, \text{ and } \text{nr}(x_s) \geq \text{nr}(x_q)\}^{33}.$$

This yields the mapping of Fig. 4-16.

The next step in the initialization exploits the following fact:

Let, for a given set of units U' , $\text{mem}_x(U')$ be the set of all centers that are elements of the tuple of some unit $\in U'$.

Then, for all $u_q \in U_q$, $x_q \in t_q$ implies $f_X(x_q) \in \text{mem}_x(F_U(u_q))$.

32) This is the only place where comparison of attribute sets is done in the algorithm, but, in contrast to Sussenguth's algorithm, this is sufficient to identify all compatible mappings correctly because the unit correspondences are also established.

33) The second constraint does not impose any restrictions on the mapping in the example, but it helps handling polycyclic ring systems.

$F_X:$		-	-	-	-		-
	-	1	1	1	1	1	1
		.	.	1	.	.	1
		1	1	1	1	1	1
		1	1	1	1	1	1
		1	1	1	1	1	1

Fig. 4-16

This leads to a restricted F_X in the following way:

```

for all  $u_q \in U_q$ 
  for all  $x_q \in t_q$ 
     $F_X(x_q) \leftarrow F_X(x_q) \cap \text{mem}_x(F_U(u_q));$ 
  endfor;
endfor;
```

For instance in the example $F_U(0_q) = \{0, 1\}_s$, thus

$$F_X(3_q) \leftarrow \{0, 1, 2, 3, 4, 5, 6\}_s \cap \{1, 5\}_s.$$

The resulting F_X is shown in Fig. 4-17.

$F_X:$		-	-	-	-		-
	-	.	.	1	1	.	1
		.	.	1	.	.	1
		1	.	.	1	.	.
		.	1	.	.	1	.
		.	.	1	.	.	.

Fig. 4-17

4.4.2.2. Refinement of Mappings

The sequence of steps described in this paragraph is iteratively applied to the matrices representing F_X and F_U until no further changes (i.e. no reductions of image sets) are achieved.

Step 1

The cardinalities of all sets $F_X(x)$ with $x \in X_q$ are computed. If $|F_X(x_q)| = 1$ for some $x_q \in X_q$, $F_X(x_q)$ is subtracted from all other image sets because $x \neq x_q$ implies $f_X(x) \neq f_X(x_q)$ for the final mapping f_X . This is repeated until no additional changes can be made. Fig. 4-18 shows the result of step 1 to the mapping F_X in the example.

$F_X:$		-	-	-	-	-
-		.	.	1	.	1
		.	.	1	.	1
		1	.	.	1	.
		.	1	.	.	1
		.	.	1	.	.

Fig. 4-18

Step 2

Let, for a given set of centers X' , $\text{mem}_u(X')$ be the set of all units that contain some center $x \in X'$ in their tuples. Then, for all $x_q \in X_q$, $x_q \in t_q$ implies $f_U(u_q) \in \text{mem}_u(F_X(x_q))$.

In the same way as in the initialization phase of the center mapping, this leads to the following restriction procedure.

```

for all  $x_q \in X_q$ 
  for all  $u_q \in \text{mem}_u(\{x_q\})$ 
     $F_U(u_q) \leftarrow F_U(u_q) \cap \text{mem}_u(F_X(x_q))$ ;
  endfor;
endfor;
```

For instance, in the example $F_X(3_q) = \{1, 5\}_s$. The units with tuples containing 3_q are $\{0, 7, 9\}_q$ and the union of the units containing 1_s and 5_s (i.e. $\text{mem}_u(F_X(3_q))$) is $\{0, 1, 11, 13, 14, 15\}_q$. By this restriction the set $F_U(7_q)$ becomes $\{11, 13, 14, 15\}_s$. Fig. 4-19 shows the effect of the complete step to the mapping F_U .

F_U :		-	-	-	-		-	-	-	-		-	-	-	-	
	-	1	1	.	.	.	.	.	.	.	.	.	.	.	.	.
		.	.	1	1	.	.	.	.	.	.	.	.	.	.	.
		.	.	1	1	.	.	.	.	.	.	.	.	.	.	.
		.	.	.	.	1	.	.	.	.	.	.	.	.	.	.
		.	.	.	.	.	1	1	.	.	.	.	.	.	.	.
	-	.	.	.	.	.	.	1	1	1	1	1	1	1	.	.
		.	.	.	.	.	.	.	.	1	.	1	.	.	.	.
		.	.	.	.	.	.	.	.	.	1	.	1	.	.	.
		.	.	.	.	.	.	.	1	.	1	.	.	.	.	.
		.	.	.	.	.	.	.	.	.	.	.	.	1	1	.

Fig. 4-19

Step 3

If a query unit $u_q = (b, A_q, t_q)$ is to map on a structure unit $u_s = (b, A_s, t_s)$, a permutation $g \in \text{FG}(b)$ must exist that satisfies

$$g(t_s) = f_X(t_q),$$

i.e. the image of the tuple of the query unit under the center mapping f_X must be some allowed permutation of the tuple of its image under the unit mapping f_U .

On the other hand, if $x_q = t_q[i]$, x_s can only be equal to $f_X(x_q)$, if there is a unit $u_s = (b_s, A_s, t_s) \in F_U(u_q)$ for which $g(t_s) \in F_X(t_q)$, especially $g(t_s)[i] \in F_X(t_q[i])$ for some $g \in FG(b_s)$. These constraints are applied simultaneously to reduce the image sets of F_X and F_U .

```

for all units  $u_q = (b_q, A_q, t_q) \in U_q$ 
  FT  $\leftarrow$  <tuple of empty sets with the same arity as  $t_q$ >
  for all units  $u_s = (b_s, A_s, t_s) \in F_U(u_q)$ 
    gfound  $\leftarrow$  FALSE;
    for all  $g \in FG(b_s)$ 
      if  $g(t_s) \in F_X(t_q)$ 
        gfound  $\leftarrow$  TRUE;
        for all positions  $i$  in  $t_q$ 
          FT[i]  $\leftarrow$  FT[i]  $\cup \{g(t_s)[i]\}$ ;
        endfor;
      endif;
    endfor;
  if not gfound
     $F_U(u_q) \leftarrow F_U(u_q) \setminus \{u_s\}$ ;
  endif;
endfor;
for all positions  $i$  in  $t_q$ 
   $F_X(t_q[i]) \leftarrow F_X(t_q[i]) \cap FT[i]$ ;
endfor;
endfor;
```

In the example, the unit 9_q leads to a reduction of some image set:

```

 $9_q = \text{ORIENT plus } 0_q \ 2_q \ 3_q \ 4_q$ 
 $F_U(9_q) = \{14, 15\}_s$ 
 $14_s = \text{ORIENT plus } 0_s \ 1_s \ 3_s \ 6_s$ 
 $15_s = \text{ORIENT plus } 2_s \ 3_s \ 4_s \ 5_s$ 
 $F_X(0_q) = \{2, 6\}_s, F_X(2_q) = \{0, 4\}_s,$ 
 $F_X(3_q) = \{1, 5\}_s, F_X(4_q) = \{3\}_s;$ 
FT  $\leftarrow (\{\}, \{\}, \{\}, \{\})$ 
```

First $\underline{9}_q \rightarrow \underline{14}_s$ is tested. Center 0_q must map on 6_s because 2_s is not $\in t_s$. Similarly, 2_q maps on 0_s , 3_q on 1_s , and 4_q on 3_s . So the only tuple in $F_X((0_q, 2_q, 3_q, 4_q))$ is $(6_s, 0_s, 1_s, 3_s)$. The permutation g that satisfies $g((0_s, 1_s, 3_s, 6_s)) = (6_s, 0_s, 1_s, 3_s)$ is $(2\ 3\ 4\ 1)$, but $(2\ 3\ 4\ 1)$ is not $\in A_4 = \text{FG}(\text{ORIENT})$. Therefore, $F_U(9_q) \leftarrow \{\underline{14}, \underline{15}\}_s \setminus \{\underline{14}\}_s$.

Testing $\underline{9}_q \rightarrow \underline{15}_s$ we get $0_q \rightarrow 2_s$, $2_q \rightarrow 4_s$, $3_q \rightarrow 5_s$, $4_q \rightarrow 3_s$. $g = (1)(3\ 4\ 2)$ satisfies $g((2_s, 3_s, 4_s, 5_s)) = (2_s, 4_s, 5_s, 3_s)$, and $g \in A_4$. FT becomes $(\{2\}_s, \{4\}_s, \{5\}_s, \{3\}_s)$, and so $F_X(0_q) \leftarrow \{2, 6\}_s \cap \{2\}_s$, $F_X(2_q) \leftarrow \{0, 4\}_s \cap \{4\}_s$, $F_X(3_q) \leftarrow \{1, 5\}_s \cap \{5\}_s$, and $F_X(4_q) \leftarrow \{3\}_s \cap \{3\}_s$. The matrix representations of F_X and F_U after this step are shown in Fig. 4-20.

$$F_X: \begin{array}{c|cccccc} & | & - & - & - & - & | & - \\ \hline - & & . & . & 1 & . & . & . & . \\ | & & . & . & 1 & . & . & . & 1 \\ | & & . & . & . & . & 1 & . & . \\ | & & . & . & . & . & . & 1 & . \\ | & & . & . & . & 1 & . & . & . \end{array}$$

$$F_U: \begin{array}{c|cccccc|cccccc|cccccc|} & | & - & - & - & - & | & - & - & - & - & | & - & - & - & - & | \\ \hline - & 1 & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ | & . & . & 1 & 1 & . & . & . & . & . & . & . & . & . & . & . & . \\ | & . & . & 1 & 1 & . & . & . & . & . & . & . & . & . & . & . & . \\ | & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . \\ | & . & . & . & . & . & 1 & 1 & . & . & . & . & . & . & . & . & . \\ - & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . \\ | & . & . & . & . & . & . & . & . & 1 & . & 1 & . & . & . & . & . \\ | & . & . & . & . & . & . & . & . & . & 1 & . & 1 & . & . & . & . \\ | & . & . & . & . & . & . & . & . & 1 & . & 1 & . & . & . & . & . \\ | & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & \end{array}$$

Fig. 4-20

Step 4

If the first three steps fail to reduce at least one of the image sets, an additional constraint is applied. This test is (computationally) rather expensive, thus it is only performed if every other means fails to improve the mappings and only for centers x_q with small image sets $F_X(x_q)$.

If a center x_q is to be mapped on a center x_s , then, for every u_q containing x_q in its tuple (i.e. for every $u_q \in \text{mem}_u(\{x_q\})$), a unit $u_s \in F_U(u_q)$ must exist that is also $\in \text{mem}_u(\{x_s\})$ and it must be possible to map each u_q on a different u_s . That means, the cardinality of $\text{mem}_u(\{x_q\})$ must be less than, or equal to, the cardinality of the intersection of the set of those units which contain x_s (i.e., $\text{mem}_u(\{x_s\})$) with $F_U(\text{mem}_u(\{x_q\}))$, the union of all image sets of all tuples containing x_q . In pseudocode:

```

for all  $x_q \in X_q$ 
  for all  $x_s \in F_X(x_q)$ 
    if  $|\text{mem}_u(\{x_q\})| > |F_U(\text{mem}_u(\{x_q\})) \cap \text{mem}_u(\{x_s\})|$ 
       $F_X(x_q) \leftarrow F_X(x_q) \setminus \{x_s\}$ ;
    endif;
  endfor;
endfor;
```

This step doesn't lead to a reduction in the size of any image set in the example, but it helps to improve F_X in more complicated problems.

These four steps are iterated until no further improvement is achieved. In the example, the algorithm terminates after another cycle and the resulting mapping is shown in Fig. 4-21.

$F_X:$		-	-	-	-		-
-		.	.	1	.	.	.
		.	.	.	.	.	1
		.	.	.	1	.	.
		.	.	.	.	1	.
		.	.	1	.	.	.
		.	.	.	1	.	.
		.	.	.	.	1	.
		.	.	.	.	.	1

$F_U:$		-	-	-	-		-	-	-	-	
-		1	.	.	.	.	.	.	.	.	.
		.	.	1	.	.	.	.	.	.	.
		.	.	.	1	.	.	.	.	.	.
		.	.	.	.	1	.	.	.	.	.
		.	.	.	.	.	1	.	.	.	.
		.	.	.	.	.	.	1	.	.	.
-		.	.	.	.	.	.	.	1	.	.
		.	.	.	.	.	.	.	.	1	.
		.	.	.	.	.	.	.	.	.	1
		.	.	.	.	.	.	.	.	.	.
		.	.	.	.	.	.	.	.	.	1

Fig. 4-21

Because all image sets are singletons and F_X and F_U are locally consistent, the final mappings f_X and f_U are compatible and read:

$$f_X = \{0_{\underline{q}} \rightarrow 2_s, 1_{\underline{q}} \rightarrow 6_s, 2_{\underline{q}} \rightarrow 4_s, 3_{\underline{q}} \rightarrow 5_s, 4_{\underline{q}} \rightarrow 3_s\}, \text{ and}$$

$$f_U = \{0_{\underline{q}} \rightarrow 0_s, 1_{\underline{q}} \rightarrow 2_s, 2_{\underline{q}} \rightarrow 3_s, 3_{\underline{q}} \rightarrow 4_s, 4_{\underline{q}} \rightarrow 6_s, 5_{\underline{q}} \rightarrow 8_s, 6_{\underline{q}} \rightarrow 12_s, 7_{\underline{q}} \rightarrow 13_s, 8_{\underline{q}} \rightarrow 9_s, 9_{\underline{q}} \rightarrow 15_s\}$$

4.4.2.3. Case Analysis

In the preceding case, the set reduction procedure was sufficient to compute unique center and unit mappings, but it was shown in the description of Sussenguth's algorithm that this needs not be the case. If the center

mapping is unique, the unit mapping must also be unique, because two units with permutationally equivalent tuples must have disjoint (and non-empty) attribute sets. Therefore, the case analysis, now to be described, only needs to resolve the ambiguities in the center mapping.

The procedure employed is very similar to what has been used in the work dealing with subgraph isomorphism testing:

- Select a center x_q with $|F_X(x_q)| > 1$.
- Analyze all cases $x_q \rightarrow x_s$, where $x_s \in F_X(x_q)$ by the following steps:
 - 1) Construct a mapping F'_X with $F'_X(x_q) = \{x_s\}$,
 and $F'_X(x) = F_X(x) \setminus \{x_s\}$ if $x \neq x_q$.
 - 2) Call the set reduction procedure.
 - 3) If F'_X has not been completely refined, recursively call the case analysis procedure.
- If all cases fail to yield mappings f_X and f_U , report 'non matching' to the calling procedure.
- If a match was found, return the list of matches.

For an information retrieval system like GM-Search, it is sufficient to stop analysis if one match has been found so as to reduce the effort necessary.

If, in the example of Fig. 4-14, the query is restricted to the chemical constitution, as shown in Fig. 4-22³⁴), the set reduction is not able to find a unique mapping.

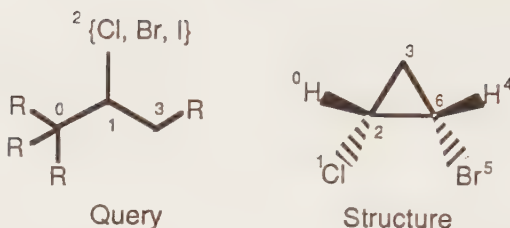


Fig. 4-22

The query GM-description Q_q is now:

units:

<u>0</u> :	IDENT	Cl Br I	H0	2
<u>1</u> :	IDENT	C	H0 H1 H2 H3	0
<u>2</u> :	IDENT	C	H1	1
<u>3</u> :	IDENT	C	H2 H3	3
<u>4</u> :	BOND	single	r3b r4b r5b r6b r7b r8b rhb chainb	0 1
<u>5</u> :	BOND	single	r3b r4b r5b r6b r7b r8b rhb chainb	1 2
<u>6</u> :	BOND	single	r3b r4b r5b r6b r7b r8b rhb chainb	1 3

Cross Reference:

- 0: 1 4
 1: 2 4 5 6
 2: 0 5
 3: 3 6

The matrix representation of the mappings F_X and F_U , after the set reduction, is shown in Fig. 4-23.

³⁴⁾ Note that the hydrogen atom at carbon 1 is described implicitly and not as an explicit atom, because it is no longer needed to express the configuration.

$F_X:$		-	-	-	-		-
	-	.	.	1	.	.	1
		.	.	1	.	.	1
		.	1	.	.	1	.
		.	.	1	.	.	.
		.	.	.	1	.	.
		.	.	.	.	1	.
		.	.	.	.	.	1

$F_U:$		-	-	-	-		-	-	-	-	
	-	1	1	.	.	.	.	.	.	.	.
		.	.	1	1	.	.	.	.	.	.
		.	.	.	1	1	.	.	.	.	.
		.	.	.	.	1	.	.	.	.	.
	-	.	.	.	.	.	1	.	.	.	.
		.	.	.	.	.	.	1	.	1	.
		.	.	.	.	.	1	.	1	.	.

Fig. 4-23

In the case analysis the set $F_X(2_q) = \{1, 5\}_s$ is used. The first case ($2_q \rightarrow 1_s$) leads to the mappings of Fig. 4-24, which, after set reduction, end up in unique mappings.

The resulting center and unit mappings are:

$$f_X = \{0_q \rightarrow 6_s, 1_q \rightarrow 2_s, 2_q \rightarrow 1_s, 3_q \rightarrow 3_s\}, \text{ and}$$

$$f_U = \{0_q \rightarrow 1_s, 1_q \rightarrow 3_s, 2_q \rightarrow 2_s, 3_q \rightarrow 4_s, 4_q \rightarrow 8_q, 5_q \rightarrow 11_s, 6_q \rightarrow 7_s\}.$$

The other case ($2_q \rightarrow 5_s$) leads to the following mappings:

$$f_X = \{0_q \rightarrow 3_s, 1_q \rightarrow 6_s, 2_q \rightarrow 5_s, 3_q \rightarrow 3_s\}, \text{ and}$$

$$f_U = \{0_q \rightarrow 0_s, 1_q \rightarrow 2_s, 2_q \rightarrow 3_s, 3_q \rightarrow 4_s, 4_q \rightarrow 8_q, 5_q \rightarrow 13_s, 6_q \rightarrow 9_s\}.$$

$F_X:$		-	-	-	-		-
-		.	.	1	.	.	1
		.	.	1	.	.	1
		.	1	.	.	.	.
		.	.	.	1	.	.

$F_U:$		-	-	-	-		-	-	-	-	
-		.	1	.	.	.	.	.	.	.	.
		.	.	1	1	.	.	.	.	.	.
		.	.	1	1	.	.	.	.	.	.
		.	.	.	.	1	.	.	.	.	.
-		.	.	.	.	.	.	1	.	.	.
		.	.	.	.	.	.	.	1	1	.
		.	.	.	.	.	1	1	.	.	.

Fig. 4-24

4.4.2.4. Improvements

Within the different cycles of the set reduction procedure, not all $F_U(u)$ have to be analyzed but only those for which the corresponding center mappings have been changed since their last examination. The implementation used in GM-Search keeps track of those changes and thus reduces effort. Additionally, only the units with a tuple of arity > 1 can contribute to the set reduction process because all information contained in the other units is exploited during initialization. Therefore, they are only updated but not examined further.

4.5. Screening

An obvious algorithm for finding all compounds in a database which contain a certain substructure is to scan through the structures and perform a

substructure test for each compound. This approach, however, is only applicable for small sets of compounds and small substructures because substructure testing is a time consuming process. To overcome this difficulty, existing chemical structure retrieval systems apply a screening phase before the actual matching is done. The idea of screening depends on the fact that being-substructure-of (as well as being-sub-GM-description-of) defines a half-ordering on the set of all structures (or GM-descriptions). When a structure is entered to the database, it is tested against a set of so-called screens, which are usually small and regularly built substructures. A query q can only be a substructure of a structure s if all screens contained in q are also a part of s . With this information, most of the structures can be excluded from being tested just by comparing the screening information without resorting to the tedious matching procedure [56].

4.5.1. Superimposed Coding

The presence or absence of a screen can be stored in a manner that simplifies this set inclusion test. One way is to set up a bit string for each compound, where bit i is set if and only if screen i is present. Another way is the construction of inverted files, where file i lists all structures containing screen i . Inverted files are advantageous if the screens are rare and the corresponding bits in the bit string would be 0 for most structures. On the other hand, the storage requirements of bit strings are better if frequent screens are to be encoded. A means to improve the storage efficiency of bit strings with rare screens is superimposed coding.

This method was invented by Mooers [58] to increase the number of attributes that could be encoded on edge-notched cards. A good description of the procedure, together with other algorithms for combinatorial hashing, can be found in [59]. The type of queries that can be answered with these "zatocoded" cards is the same as those occurring in the screening of substructure queries: Which cards bear all of a given subset of the

defined attributes? The key to space efficiency is to allow a small number of so-called "false-drops", i.e. answers not bearing all the attributes required. Instead of reserving one bit (or notch) for each attribute, they are encoded by simultaneously setting a combination of bits. The inclusive query mentioned above would be answered by searching for all cards that bear the union of all sets of notches for the required attributes. If the probability of a notch is 50% at each position and the query contains n bits, then the probability of a false-drop is $1/2^n$ while all cards with the desired attribute combination are retrieved.

In the context of substructure searching, the occurrence of false-drops is no problem because the screens do not completely describe the answer set anyway.

Mooers used a fixed number of bits for all attributes, independently of their probability, which is useful if the probabilities can be assumed to be similar. In chemical substructure searching, it is nearly impossible to devise an efficient set of screen substructures with nearly equal probability. Feldman and Hodes [60,61] extended Mooers' approach by allowing variable numbers of bits. In addition, they also exploited the mutual dependency of screens to reduce the number of bits required. This scheme will be described here in more detail because it was also applied to the encoding of the screens used in GM-Search.

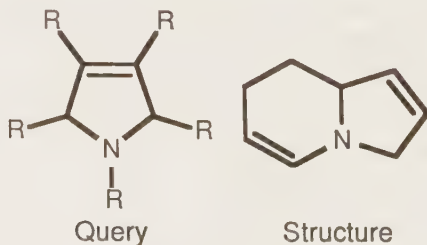


Fig. 4-25

Fig. 4-25 shows a query and a structure, for which the screening information is collected as an example. For simplicity of presentation, only so-called atom-bond-sequence screens are taken into account. Fig. 4-26 lists some possibly relevant screens³⁵⁾.

	substructure	incidence i	$\log_2 1/i$	parent	$\log_2(i_p/i)$	bit pattern	
1)	N-C-C	28.95 %	1.79	-	1.79	0000000000001001	
2)	C-N-C	26.10 %	1.94	-	1.94	0000000000011000	
3)	N-C-C=C	9.86 %	3.34	1)	1.55	1100000000000000	
4)	N-C=C	8.43 %	3.57	-	3.57	0100000011000100	
5)	N-C=C-C	7.32 %	3.77	4)	0.20	0000000000000000	
6)	C=C-N-C-C	2.18 %	5.52	4)	1.95	0000001000000001	
7)	N-C-C=C-C-O	0.92 %	6.76	3)	3.42	0010000001000010	
	structure	1) OR 2) OR 3) OR 4) OR 5) OR 6)					1110001011011111
	query	1) OR 2) OR 3) OR 7)					1110000001011011

Fig. 4-26

They are not taken from the screen set used in GM-Search, but their incidences are computed from the sample set of molecules used in the evaluation of GM-Search.

In assigning screen bit patterns to screens, n , the number of the bits to be used, can be chosen to reflect the frequency of the screen in the database. Feldman and Hodes chose this number to be $n = \log_2(1/i)$. The third column in Fig. 4-26 shows those quantities³⁶⁾. Then, n would represent the amount of information contained in the screens if they were indepen-

35) The screen N-C=C, for instance, describes a substructure with a nitrogen atom bonded to a carbon atom by a single bond, and this carbon atom is connected to another carbon with a double bond.

36) The actual bit numbers are rounded off to the nearest integer.

dent. This is obviously not the case. For instance, if screen 6) is present in a molecule, screen 4) must also be present. Thus, Feldman and Hodes refined their formula to $n = \log_2(i_p/i)$, where i_p is the frequency of the least frequent screen implied by the screen in question (and which has any new bits set on its own).

The corresponding parents are indicated in column 4 of Fig. 4-26 and the values of $\log_2(i_p/i)$ are given in column 5. The bit pattern for screens with $n > 0$ is computed by copying all bits of the parents and randomly setting n additional bits. Set bits are denoted by '1' while '0' is used for non-set bits. The bit pattern for structure and query are computed by simply 'OR'-ing the bit patterns of the appropriate screens. A structure s passes the screening test for being superstructure of a query q if all bits set in the bit pattern of q are also set in the pattern of s, as is the case in the example of Fig. 4-25.

Feldman and Hodes report [60] that, by this scheme, they could encode 3195 screens in as few as 96 bits in the WRAIR substructure search system. Their screen substructures were selected algorithmically according to statistics acquired from a sample of the whole WRAIR file; they include chains, rings, and singly branched fragments of up to 11 atoms.

4.5.2. Screen Types Used in GM-Search

The screens used in GM-Search are divided into two classes, namely matching screens and non-matching screens. This distinction arises from the way their presence or non-presence in a structure or a query is determined. Matching screens are handled by the ordinary sub-GM-testing procedure described above whereas special purpose algorithms are used for the non-matching screens. This section describes the various classes of non-matching screens used in GM-Search.

4.5.3. Element-Count Screens

If a query is a substructure of a structure, the number of atoms of a certain atom type in the query must be less than, or equal to, the number of atoms of the same atom type in the structure. Therefore, in most, if not all, substructure search systems with screening, element-count screens are used. Such a screen will be denoted by

EC n AS,

where EC is the screen type descriptor, and n is the number of atoms with atom types from some atom type set AS.

To exploit the extended superimposed coding method by Feldman and Hodes, frequencies of occurrence for the screens are needed. The number of molecules collected in the experimental database is too small to give realistic estimates for the frequencies of rare elements in a production system such as CAS-ONLINE. Therefore, the probabilities were taken from the CAS-ONLINE Screen Dictionary [7], which also contains element-count screens.

For the more common elements, such as carbon, oxygen, nitrogen, and the halogens, more than one screen is used to describe the number of atoms of that type contained in a structure. For example, for boron four screens are allocated:

number	screen	frequency	parent	number of bits
1923	EC 1 Al B Ga In Tl	1.95 %	-	6
1932	EC 1 B	1.02 %	1923	1
1933	EC 2 B	0.14 %	1932	3
1934	EC 3 B	0.06 %	1933	1

This means, for instance, that 1.95% of the compounds registered at CA in 1981 contained at least one atom with an atom type from the set {Al, B, Ga, In, Tl}. The first column lists the screen numbers from the

CAS-ONLINE screen dictionary. The last two columns contain the parent screen and the number of bits allocated for that screen in GM-Search, respectively³⁷⁾. For the rarer elements, it is assumed that only few of them occur in the same molecule. Therefore, no individual screens but combinations of screens are used for the columns and the rows of the periodic table. In GM-Search, 95 element-count screens are used, which are mostly directly taken from the CAS-ONLINE screen dictionary, augmented with some combined screens to reduce the number of bits assigned to each individual screen.

4.5.4. Type-of-Ring Screens

The second kind of screens in GM-Search deals with the ring systems present in a structure. Since it is impossible to supply screens for every conceivable ring system and subsystem, the description scheme used for the CAS-ONLINE screens was adopted here to represent rings and their fusion patterns.

The cholestane ring system depicted in Fig. 4-27 has the following so-called type-of-ring screens:

³⁷⁾ The table also gives an illustration of the space saving by superimposed coding. In CAS-ONLINE, four bit positions within their 2,000 bits long screen bit string are occupied by those four screens, and for most of the compounds (98.05%) these bits are zero. Assuming that, for a bit in the final superimposed code, 50% of all structures in the database have this bit set, the average number of bits that have to be allocated for the four element-count screens is about $2 \times (6 \times 1.95\% + 1 \times 1.02\% + 3 \times 0.14\% + 1 \times 0.06\%) = 0.264$ bits.



Fig. 4-27

number	screen	frequency
1859	TR 1 DDDTT	14.86 %
1874	TR 1 DDDDTT	27.59 %
1885	TR 1 DDTTTT	6.53 %
1886	TR 2 DDTTTT	3.71 %

As in the case of element-count screens, the first two characters tell the screen type (i.e. type-of-ring) followed by the minimal number of occurrences of the screen in the structure. The string of 'T'-s and 'D'-s stands for the ring fusion pattern. A 'T' means a bridge head atom, whereas a 'D' is used for an ordinary ring atom.

In CAS-Online, TR-screens are only used if a ring system is declared to be isolated, i.e., no other ring is fused to it and it bears no spiro junction. This usage has the advantage that TR-screens are very effective if they may be used, but it also restricts their applicability.

GM-Search uses a different approach. Being a bridgehead atom is more specific than being a ring atom. Thus, if the ring systems are not restricted to isolated ones, for a TR-screen a with n 'T'-s in its string, all other TR-screens with less than n 'T'-s on the same positions (up to ring automorphisms) must also be considered to be present.

Therefore, GM-Search associates the following TR-screens with the cholestane skeleton³⁸⁾:

number	screen	frequency	number of bits	parent
4)	TR 1 DDDDD	38.67 %	1	-
7)	TR 2 DDDDDD	45.12 %	1	-
8)	TR 3 DDDDDD	11.04 %	2	7)
15)	TR 1 TDDDD	24.82 %	1	4)
16)	TR 1 TDDDDD	45.33 %	1	-
17)	TR 1 TDDTDD	13.43 %	2	16)
21)	TR 1 TDTDDD	16.13 %	1	16)
22)	TR 2 TDTDDD	5.76 %	1	21)
30)	TR 2 TTDDDD	19.99 %	1	7)
31)	TR 3 TTDDDD	2.24 %	2	30)
40)	TR 1 TTTDDD	10.67 %	1	21)
45)	TR 2 TTTTDD	3.71 %	1	22)

The frequencies of the screens are derived from those published in [7] for isolated ring systems by adding the frequency of a TR-screen a to all screens that imply a. This approximation (i.e. mutual exclusion of screens) works fairly well and is easier to implement than the assumption of independence, which may be more realistic. The frequencies of the type-of-ring screens that map to the same screen bit in [7] (and for which, therefore, only one frequency is given) are assumed to contribute the same amount each to the total frequency.

³⁸⁾ Among the different strings which describe the same fusion pattern for a ring, the lexicographically largest one was selected. CAS uses the lexicographically smallest one.

The presence of the TR-screens is tested by generating all sequences of ring bonds with length ≤ 7 , selecting rings, identifying bridgeheads (atoms with more than two ring bonds), building the corresponding screen string, and setting up all possible parents. The lexicographically sorted list of these ring patterns is compared with the list of TR-screens and the corresponding bits are set in the screen bit string.

4.5.5. Atom-Bond-Sequence screens

The last kind of non-matching screens describes atom-type/bond-type sequences. Because GM-Search allows queries with generic atom and bond types, five different levels of specificity are defined for those screens:

- Bond sequences:

All different atom types (and atom type sets) are regarded equal and only the bond type information distinguishes the screens. They are written, for instance, as AB 1 A-A=A#A≈A, where '-', '=', '#', and '≈' mean single, double, triple, and aromatic bond types, respectively.

- Carbon/heteroatom sequences:

All bond types are regarded equal and the atom types are classified as carbon and non carbon (hetero) atoms. They are written like AB 1 Q?C?C?Q?Q, where 'C' is a carbon and 'Q' is a heteroatom.

- Carbon/heteroatom-bond sequences:

This type carries the combined information of the two types above. Atoms are classified as carbon and heteroatoms and bond orders are also differentiated.

- Atom sequences:

They describe sequences of atoms and classify them more than just in carbon and heteroatoms. The following classes are used:

C for carbon,
O for oxygen,
N for nitrogen,
X for halogen,
S for chalcogenes except oxygen,
M for metals,
P for phosphorus and arsenic, and
Y for all other non metals.

An example is the screen AB 1 N?C?C?P?M

- Atom-bond sequences:

They use the same atom type classes as the atom sequence screens and additionally specify the bond types of the bonds between the atoms.

Some further remarks have to be made:

- 1) For any of these screen classes, a terminal '0' in the screen string means a cyclic structure.
- 2) Bonds and atoms which do not fit completely into one of the classes defined for a certain screen type are not part of a screen of this type.
- 3) Screens containing only carbon atoms or consisting of only one atom occur in more than one class. They are, therefore, generated more than once, but only the first occurrence is recorded. (In atom-bond sequence screens, the repeat count is always set to one, because otherwise too many different screen classes have to be considered.)
- 4) To avoid generating excessive screen candidates, the length of the screen strings is restricted:

sequence of	max. chain length	max. ring size
bonds	6	7
carbon/heteroatoms	6	7
carbon/heteroatoms-bonds	6	0
atoms	4	7
atoms-bonds	4	0

The actual AB-screen set used in GM-Search was selected by:

- 1) Exhaustive generation of all screens conforming to the above stated constraints for all compounds in the sample structure library.
- 2) Computing the screen frequencies by sorting the generated screens and counting the equivalent ones.
- 3) Evaluating which parent screen contains which descendant.
- 4) Selecting the screens that receive non-zero bit counts according to the procedure by Feldman and Hodes [60].

The final set contains 1086 different AB-screens.

The procedure of finding the AB-screens for a given molecule or query is similar to the one finding of TR-screens:

All sequences of atoms and bonds with less than 7 atoms (the maximum number of centers for all AB-screen types) are generated. Then, for every AB-screen type, the atoms and bonds in each sequence are classified and the corresponding AB-screen descriptors are accumulated. The final set of AB-screen descriptors is sorted and then compared with the (also sorted) list of actually defined screens. The bit patterns of the included screens are then set in the screen bit string.

4.5.6. Matching Screen Generation

The efficiency of a screening system for substructure search does not only depend on its own structure, but also on the queries. Therefore, GM-Search uses an additional screening system to adapt to the expected queries. Another reason was that there is no obvious way for automatic generation of screens containing stereochemical information. Thus, they were left to be part of the set of matching screens. The screens that are called 'augmented atoms' in other systems like CAS ONLINE are also contained in this subsystem.

There are three occasions, when matching screens have to be considered:

- 1) When a new structure is added to the database.
- 2) When a substructure query is given to the system.
- 3) When a new screen is added to the set of matching screens.

The cases 1) and 2) are so similar, that they will be discussed together. The set of matching screens is organized in the same way as the set of molecules comprising the structure library, except that an additional bit string is associated with each screen. The relation 'being sub-GM-description of' defines a half ordering on the screens. Thus, it is possible to arrange them in such a way that, if one matching screen a is a sub-GM-description of some other screen b, the screen a has a lower ordinal number than b in the library of matching screens. Keeping this in mind the cases 1) and 2) can be described as follows.

First, the non-matching screens contained in the structure (query) are computed. With this information, the matching screens are examined sequentially. If the screen bit string of the matching screen is not contained in the screen bit string of the structure (query), accumulated so

far, it cannot be part of the structure (query) and is skipped. Otherwise, the structure (query) is tested for the presence of the matching screen substructure. If a match is encountered, those bits that are set in the additional bit string associated with the matching screen are also set in the screen bit string of the structure.

This scheme ensures that the information gathered with earlier matching screens is used to reduce the number further sub-GM-tests. After all matching screens have been processed, the final screen bit string of the structure is stored with it in the database. In the case of the search for a query, the final screen bit string can be used to screen out obvious non matches in the search.

The case 3) is more involved. If the new screen is part of some of those already in the matching screens library, the whole database has to be rebuilt and the matching screens must be added to their library in the proper order. Because modifying the screening system is rare and several additions can be batched to be run during a single update, this is not a serious problem.

If the new screen is not part of any matching screen already in use³⁹⁾, the procedure is as follows:

- Perform a substructure search for y, the matching screen substructure, i.e. find the non-matching screens present, check the matching screens defined so far, set the appropriate bits in the screen bit string for y, and test all candidate molecules passing the screening if they contain y.

³⁹⁾ This is also the case for each matching screen when the screen library is rebuilt in the proper order.

- Compute the percentage x of bits set in the screen bit strings of all candidates of the search.
- If m and c are the numbers of matches and candidates, respectively, assign $n = \text{round}(\ln(m/(c-m))/\ln(x))$ bits to the new screen⁴⁰.
- If n is less than one, cancel this screen, because it wouldn't improve the screen set, otherwise proceed.
- Randomly select n bit positions (not already present in the screen bit string of y) in the additional bit string for y .
- Set the corresponding bits and add the bit string and the screen to the library of matching screen.
- Go through the library of structures and update the screen bit strings of all molecules that previously matched y by setting the bits set in the additional bit string for y .

Fig. 4-28 shows some of the presently used matching screens in GM-Search.

⁴⁰) The formula is derived to fulfill the condition that, on the average, the new matching screen should reduce the number of false candidates to approximately the number of actual matches.

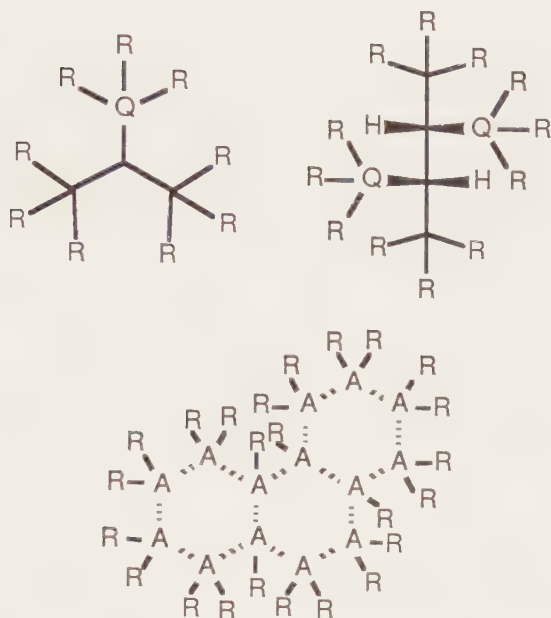


Fig. 4-28

5. Implementation

The structure retrieval system GM-Search developed in this thesis consists of three major parts: The Library Adder, the Library Searcher, and the Screen Managers. The general design of these components and the file structure used are outlined in this chapter.

5.1. The File Structure

In GM-Search, a structure library consists of three files as depicted on the right of Fig. 5-1. Their names are constructed by appending the extensions '.IND', '.BIN', and '.TEX' to the name of the library.

The library contains the following pieces of information for each structure:

- 1) The compressed binary code of the canonized GM-description.
(variable length)
- 2) The text string associated with the structure.
(variable length)
- 3) The screen bit string.
(96 bits = 12 bytes)
- 4) A hash code computed from the canonized GM-description.
(32 bits = 4 bytes)

The two variable length parts 1) and 2) are stored in the '.BIN'- and the '.TEX'-files, respectively. The '.IND'-file contains indexing information for these files, plus screen bit string, and hash code.

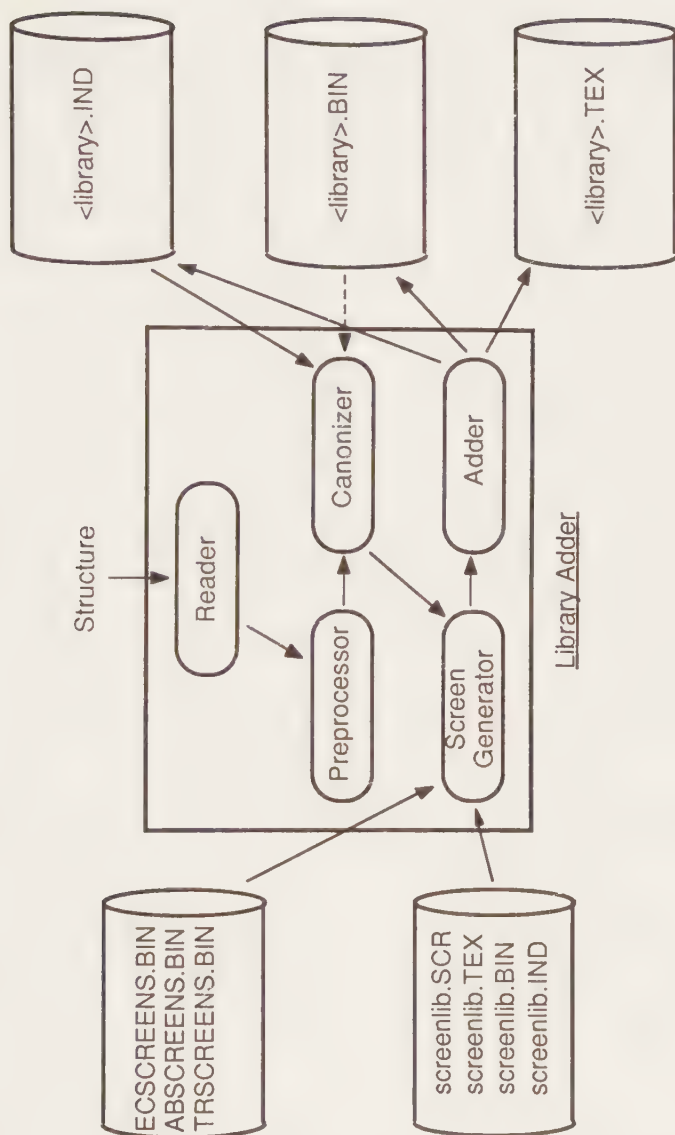


Fig. 5-1

The left side of Fig. 5-1 shows the file structure for the screens. The files for the element-count, type-of-ring, and atom-bond-sequence screens are combined in one symbol, as are the files of the library of matching screens below them. This library has the same general structure as a library of molecules except that an additional '.SCR'-file is used, which contains the screen bits to be set if the corresponding screen substructure is part of the structure being processed.

5.2. The Library Adder

When the Library Adder processes a molecule, it performs the following operations:

- The text GM-description of the molecule (written by the TO LIBRARY command in the FRAME program) is read and translated to an internal binary GM-description.
- This GM-description is preprocessed by pruning and collecting ring size information.
- The result is canonized, i.e. a unique numbering is computed, the GM-description renumbered, and a 32 bit hash code is derived therefrom.
- The '.IND'-file is checked for any structure with the same hash code⁴¹⁾ and if there are any, the corresponding GM-descriptions are read from the '.BIN'-file and tested for equality with the structure to be added.
- If the structure is not yet in the library, the screen bit string is computed by testing for the defined non-matching and matching screen substructures and setting the appropriate bits.

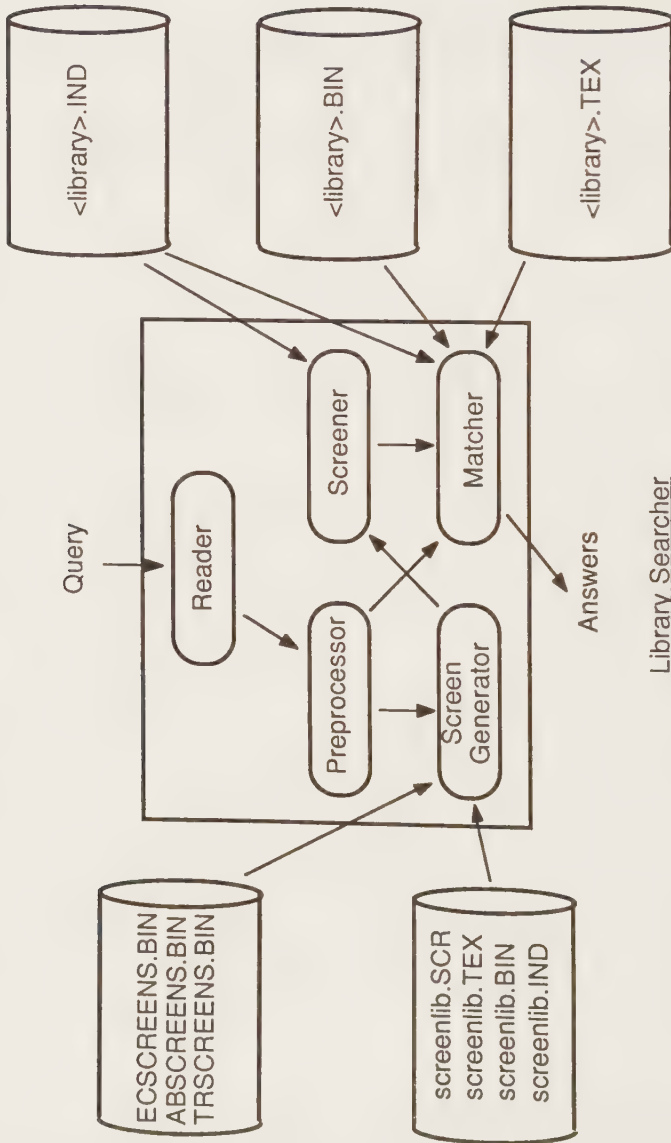
- The canonically renumbered GM-description is compressed into a sequence of bytes for efficient storage.
- Finally the byte sequence, the text string, and the index structure are appended to the '.BIN'-, '.TEX'- and '.IND'-files.

5.3. The Library Searcher

Fig. 5-2 shows the information flow of the Library Searcher. A query posed to the Library Searcher is processed in the following way:

- The text GM-description is read and preprocessed as in the Library Adder.
- The screen bits, corresponding to the screen substructures contained in the query, are set in its screen bit string.
- The Screener reads the '.IND'-file of the library and collects those structures that have a screen bit string in which all bits, which are set in the query screen bit string, are also set.
- All these candidate structures are then compared against the query by the Matcher, and the text lines corresponding to the structures containing the query are printed.

41) Part of the hash code could have been used to partition the '.IND'-file to speed up access [62], but this was not worthwhile in our case, because of the small library used.



Library Searcher

Fig. 5-2

5.4. The Screen Managers

Feldman and Hodes [60] showed that the efficiency of the superimposed codes of a set of screen substructures strongly depends on the set of structures to be handled. Therefore, the third component of GM-search maintains the set of screen substructures and their bit strings.

The element-count and type-of-ring screens and their frequencies are derived from the CAS-Online Screen Dictionary [7] and therefore their bit strings stay fixed. On the other hand, the atom-bond-sequence screens (or AB-screens) and the matching screens are made dependent on the structure library. The necessary statistics are computed from the whole library because it is small. With a larger set of structures an appropriate sample had to be selected. Fig. 5-3 describes the Screen Manager for matching screens.

This Screen Manager reads a matching-screen substructure, preprocesses it, and generates its screen bit string from the old screen set. Then it collects the candidates from the library and matches them against the screen substructure to be added. This procedure is the same as with the Library Searcher.

If the screening precision (i.e. the number of matches divided by the number of candidates) is too high, the new matching screen is discarded as unnecessary. If this is not the case, the bit pattern for this matching screen is determined, the screen substructure is added to the library of matching screens, and the new screen bit string is appended to the '.SCR'-file.

The mode of operation of the AB-Screen Manager is depicted in Fig. 5-4. The Sequence Generator reads the structures of the library and writes all those corresponding AB-sequences to the sequence file that conform to the restrictions given in subsection 4.5.5. This file is then sorted and the

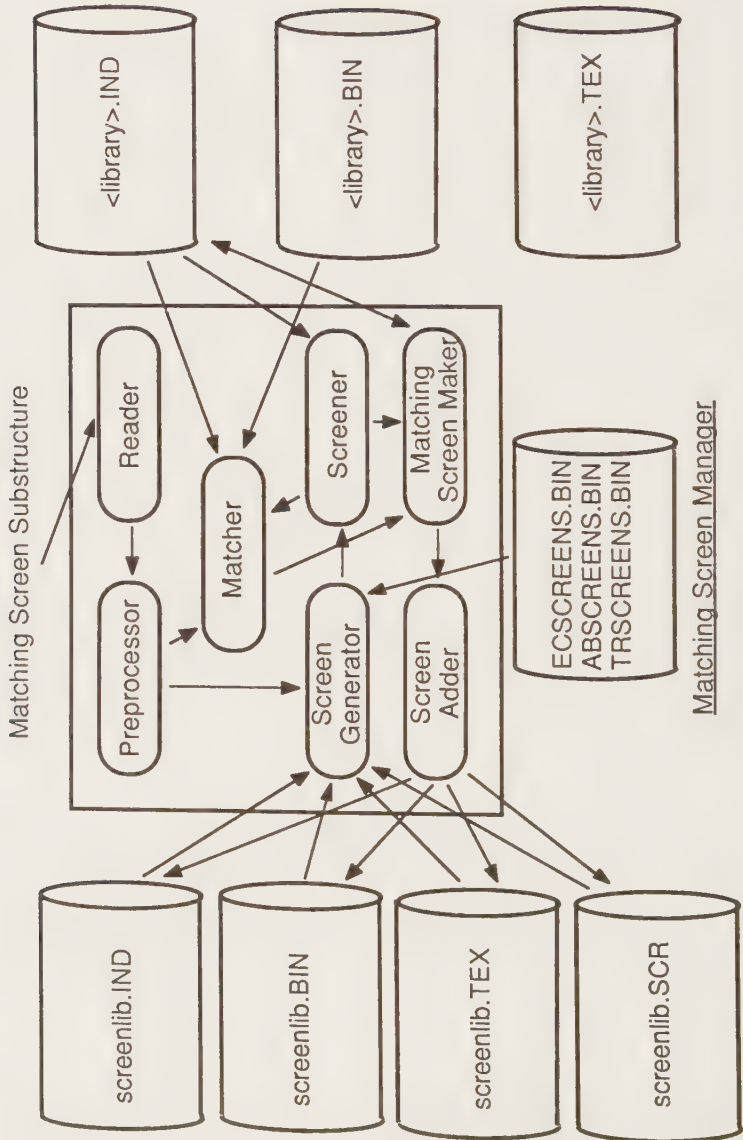


Fig. 5-3

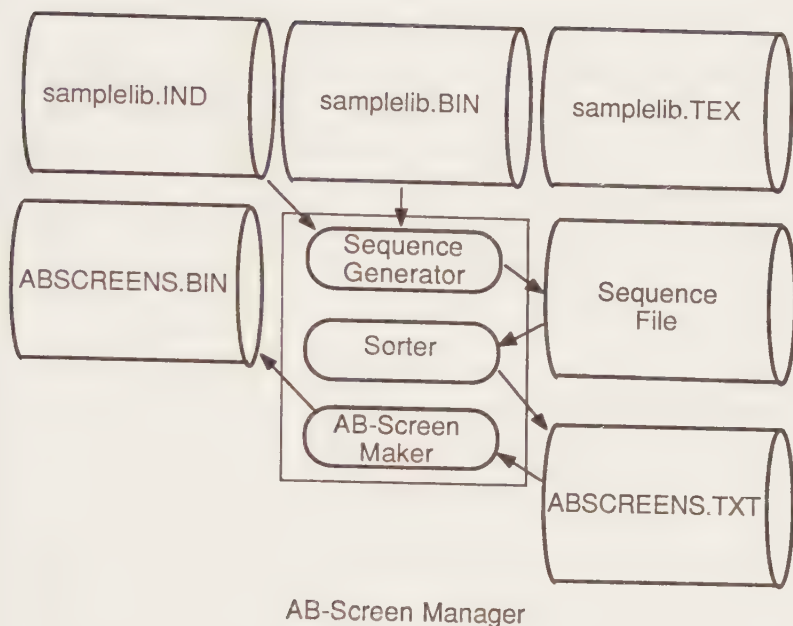


Fig. 5-4

result handed to the AB-Screen Maker. The AB-Screen Maker counts the frequencies of the sequences and determines which of them is implied by which. The rules of subsection 4.5.5 are used to calculate the number of bits in the screen bit string for each screen. The corresponding bit pattern is assigned to the screens with a non-zero bit count. The array of AB-screen descriptions (sorted by string and by number of occurrence) is then output to the file 'abscreens.BIN'. Different from the case of matching screens, the entire structure library has to be rebuilt from the text descriptions of the structures, when a new set of AB-screens has been computed, because all the bit positions in the screen bit string might have been affected.

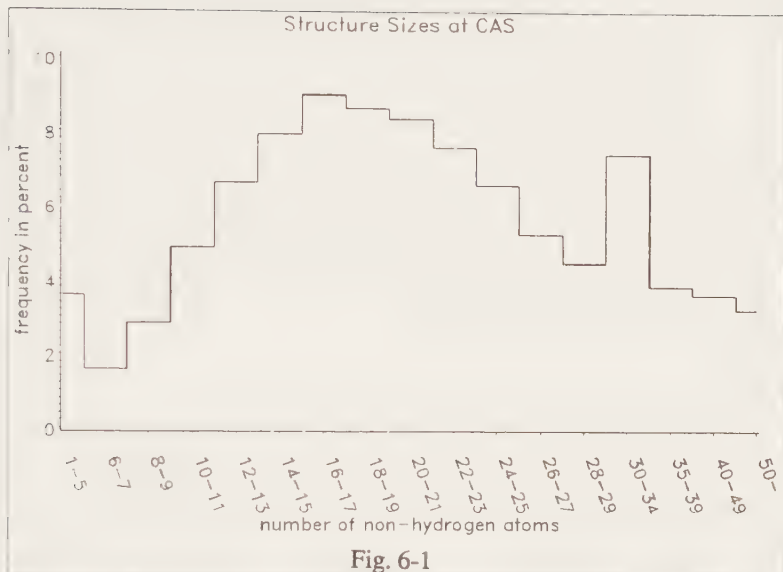
6. Performance

The complete information retrieval system GM-Search, which consists of graphical input of structures and queries as well as the programs for augmenting and searching the data base, has been implemented on a graphics workstation PERQ-2. The non-graphical parts were also ported to an IBM-PC/AT-2 (or AT for short). The canonization procedure was also tested on a DEC-10 mainframe computer, which is familiar to many chemists working on computer applications in chemistry.

The implementation on the AT has been chosen as a frame of reference in this discussion of performance for the following reasons:

- The AT is much more common than the PERQ-2 workstation.
- It is a state-of-the-art personal computer.
- The graphical interface used to input the structures and queries on the PERQ-2 is not portable, because it makes use of PERQ-specific features. It is, therefore, neglected in this discussion.
- Structures and queries are transferred as text files from the input program to the programs handling the structure library; thus there is a clean interface to some other graphical or non-graphical structure input program.

For the (over 7 million) chemical structures stored at CAS, statistical data is available on the number of structures as a function of the number of non-hydrogen atoms they contain. Fig. 6-1 shows these frequencies graphically.



This information can be utilized to estimate the time required for certain operations on an average chemical structure using mean values derived from the structure library used in this evaluation. These estimates will be given in the following whenever possible.

6.1. Comparison of PERQ-2, IBM-PC/AT-2, and DEC-10

Although the AT has been chosen as the reference computer, it is interesting to compare its speed with the two other machines. The canonization algorithm was selected for this purpose, and a driver program was written to test the procedure on certain classes of regular graphs⁴²⁾. The following list describes the classes of graphs adopted for comparison:

⁴²⁾ GM-descriptions can be used to describe graphs and graph classes are easy to define.

n-Rings

Only BOND-units are used in the description and an n-membered ring (n-ring) has a GM-description with n units. The canonization of an n-ring yield two generators.

n-Clouds

A collection of n not connected atoms is described by n IDENT-units with {He} as the atom type set. The canonization of an n-cloud yield n-1 generators.

n-Cubes

An n-cube is defined to have 2^n nodes numbered from 0 to 2^n-1 and two nodes are bonded if the binary representation of their node number differs only in one digit. Fig. 6-2 shows the graphs of 1-, 2-, and 3-cubes numbered according to this rule. The description of an n-cube consists of $2^{n-1} \times n$ BOND-units and the generating set of its automorphism group contains n generators.

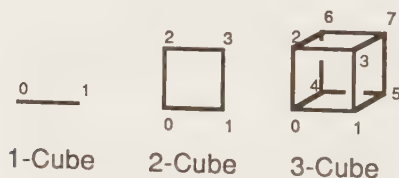
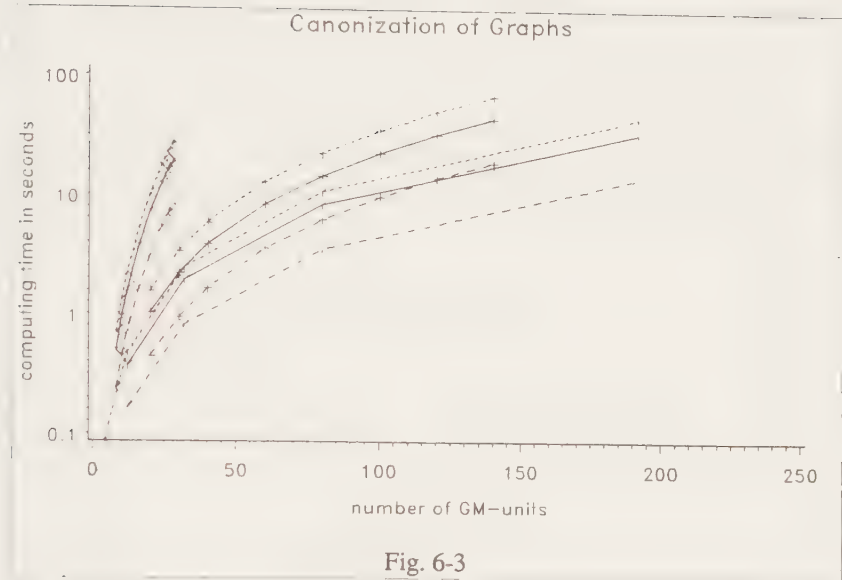


Fig. 6-2

The diagram of Fig. 6-3 shows the dependence of the canonization time on the number of units of the GM-description for the different graph classes on the three computers. Rings, clouds, and cubes are denoted by the symbols 'x', '+', and 'Y', respectively. The times on the AT are connected by solid, those on the PERQ-2 by dotted, and those on the DEC-10 by dashed lines.



The timing data can be approximated by a simple function of the form $t = a \times u^b$, where t is the computing time and u is the number of units in the GM-description. The parameters a and b have been fitted by non-linear least squares analysis. For the rings b is close to 2, for the clouds b approaches 3, and for the cubes $b = 1.63$ results in a good fit. The quality of fit can be seen on Fig. 6-4, which shows the quotient t/u^b as a function of the number of units.

Fixing b to these values the size of a can be used to compare the speed of a certain computer. The results are summarized in Table V-1.

From these values, it can be concluded that, for the implementation of the canonization algorithm in C, the DEC-10 is about 2.4 times as fast as an AT and that the PERQ-2 is 1.5 time slower than an AT.

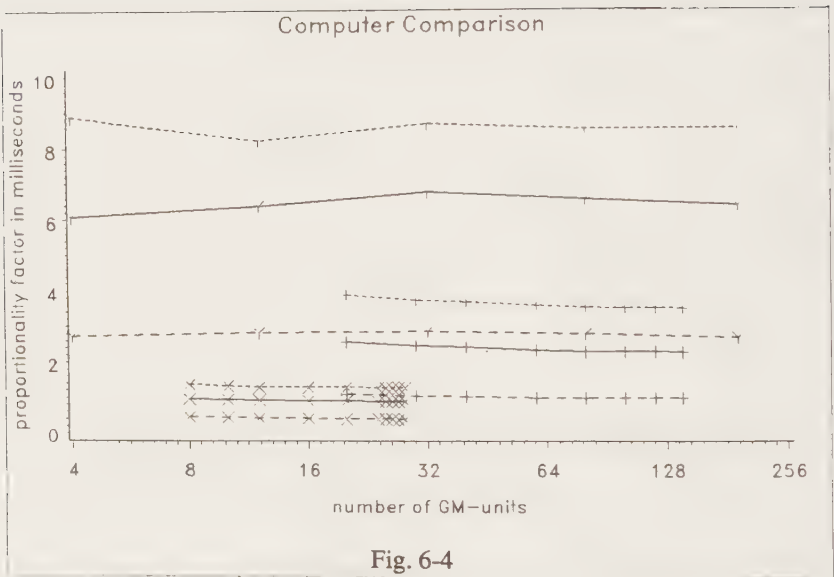


Table V-1

	ring	cloud	cube
DEC-10	$0.98 \times u^2$	$0.38 \times u^3$	$2.77 \times u^{1.63}$
IBM-PC/AT-2	$2.34 \times u^2$	$0.90 \times u^3$	$6.42 \times u^{1.63}$
PERQ-2	$3.53 \times u^2$	$1.25 \times u^3$	$8.56 \times u^{1.63}$

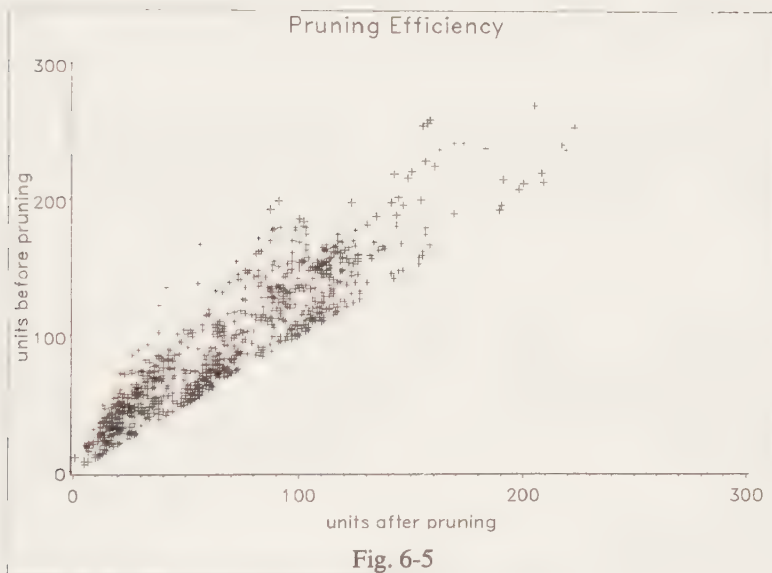
As the other non-input-related algorithms in GM-Search are also implemented in C, these figures can be taken as estimates for the performance of an implementation of GM-Search on any of the three machines.

6.2. Pruning

The performance analysis of the pruning step has to consider two values:

First, the pruning efficiency, i.e. the number of explicit atoms saved by pruning, and second, the computing time required for this step.

It was mentioned in the section on the pruning algorithm that in GM-Search pruning cannot be as efficient as in systems not handling stereochemistry. Fig. 6-5 is a scatter diagram plotting the number of units in the GM-description before pruning vs. after pruning.



Each 'x' represents a molecule of the library. It can be seen that some points are close to the main diagonal, which means that efficiency is low, while others are far above. Because hydrogen counts are not available for the CAS structure file, pruning efficiency cannot be extrapolated, but, for the sample library, the average reduction in the number of units is 28 %, while the number of centers is reduced by 34 %.

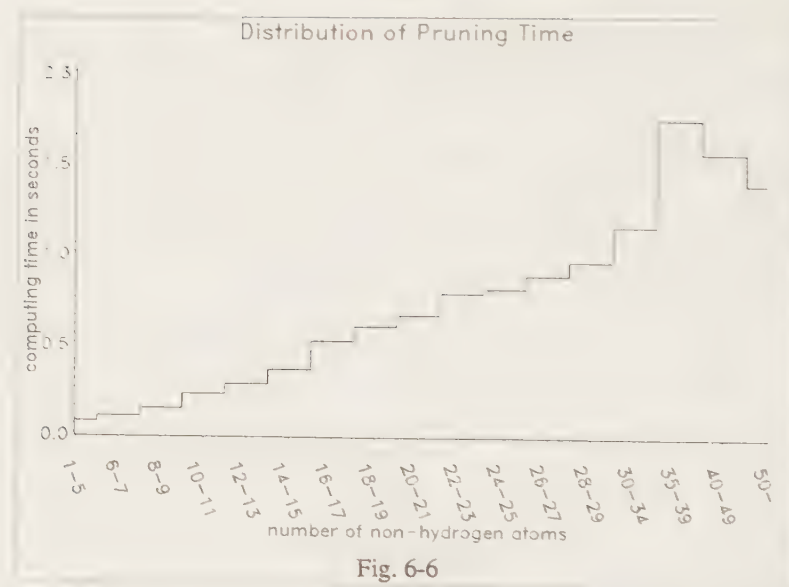


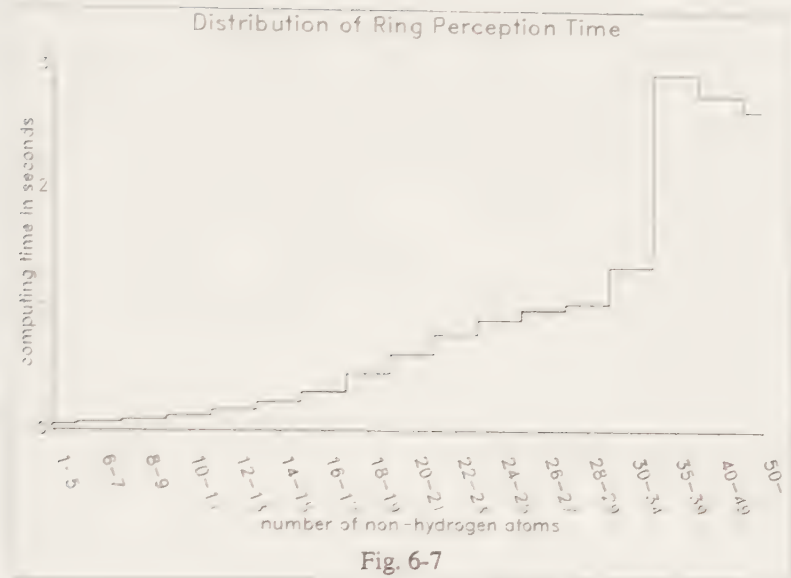
Fig. 6-6

Fig. 6-6 plots the average pruning time as a function of the number of non-hydrogen atoms in the structure. Using these values, together with the frequency data from the CAS structure file, the pruning time for an average molecule can be estimated to be 0.66 s on an IBM-PC AT-2⁴³).

6.3. Ring Perception

If a structure is added to the library, the ring size information for the bonds is computed. The distribution of the computing time required for this operation is shown in Fig. 6-7. The average ring perception time can be computed in the same way as for the pruning step, resulting in an estimate of 0.73 s for an average chemical structure.

⁴³) The value is computed by adding the products <frequency of range at CAS> times <mean pruning time for that range in GM-Search>.

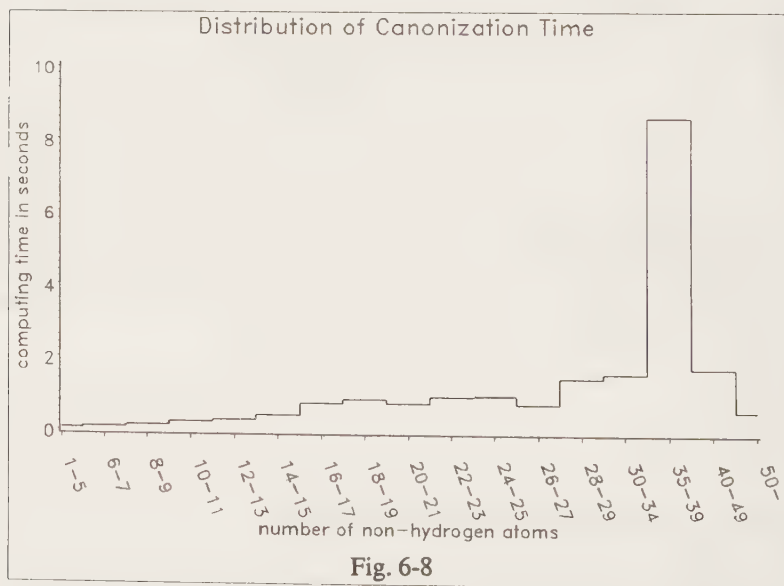


If the computing time is related to the total number of atoms, instead of the number of non-hydrogen atoms, it shows a quadratic dependence: $\langle \text{ring perception time} \rangle \approx 0.81 \times N^2$ [ms], where N is the number of centers in the GM-description.

6.4. Canonizing

Part of the performance characteristics of the canonization algorithm can be gathered from the section on computer comparison in this chapter. If the refinement procedure succeeds in finding the automorphism partition, as is the case in the sample graphs and in presumably all chemical structures, the running time of the canonization algorithm is bounded by a polynomial in the number of units in the GM-description. This can be proven as follows:

The recursion depth is bounded by n , the number of centers in the GM-description. If 'refine' always finds the automorphism partition, each code analyzed, except the first one, gives rise to a new generator of the automorphism group. Since every generator shows the equivalence of at least two nodes, and since $n-1$ generators are sufficient to prove all nodes equivalent, at most n codes will be generated. The operations in the refinement step and on each recursive level of 'canonize' can be completed in polynomial time; thus the time for the whole process is bounded by a polynomial in the number of units of the GM-description being canonized.



The worst cases for GM-descriptions of molecular structures occur if the molecule contains many phenyl substituents. Each of them increases the number of generators by one and therefore causes a comparison of codes (renumbered GM-descriptions). It also adds 47 (11 IDENT, 12 BOND, and 24 DIHED) units to the GM-description, while contributing only 6 non-hydrogen atoms.

This is the reason for the peak at 30-35 non-hydrogen atoms in Fig. 6-8, which shows the distribution of computation time for canonization. The cited range contains 6 compounds with 4 phenyl groups each and 5 to 7 generators. Their canonization time ranges from 12 to 21 seconds. Despite this irregularity, the canonization time for an average chemical structure is estimated to be 1.03 s.

6.5. Screen Generation

The most expensive step in processing a structure to be added to the library is the generation of screens. In 1981, CAS processed 35,000 substance entries per week, but only 7,000 of them occurring for the first time [6]. Therefore, screen generation had to be done only for every fifth entry.

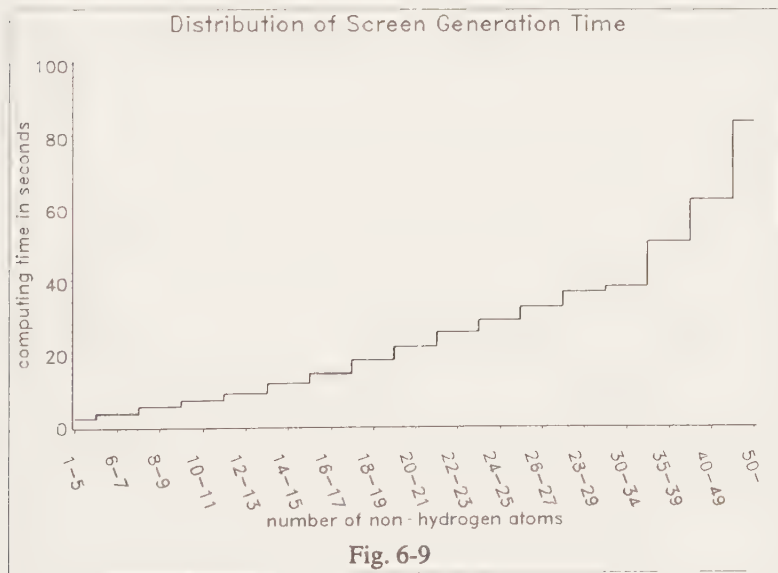


Fig. 6-9 shows the distribution of screen generation time with the number of non-hydrogen atoms. These values result in an estimate of 22.74 seconds for screen generation for an average chemical structure and 4.55 seconds for an average entry.

6.6. Searching

The cost of a substructure search is determined by the cost of different steps:

- Preprocessing the query, i.e. pruning, ring perception, and screen generation.
- Reading the screen bit strings of the structures and comparing them with the query screen bit string.
- Matching the candidate structures with the query.

The last step is also influenced by the number of candidates and by the percentage of matches among them.

At CAS [5], substructure queries are processed by a network of computers. An IBM mainframe computer performs the dialog with the user, generates screens, and deals with the text database. After preprocessing by the mainframe, the query is handed to a set of 10 pairs of PDP 11/45 and PDP 11/44 minicomputers, which perform the actual search and return the so-called registry numbers of the answers. The PDP 11/45s are dedicated to the comparison of screen bit strings and the PDP 11/44s match the resulting candidates with the query. Each pair of minicomputers has to handle 750,000 structures and one 300 MByte disk drive is allotted to the storage of screen bit strings and one to the structure descriptions. 20,000 structures are allowed to pass the screening in a substructure search, and

the number of matches is restricted to 5,000. The search is completed in approx. 5 minutes. That means that for a worst case query $5,000/10 = 500$ successful and 1,500 unsuccessful substructure tests have to be performed by each PDP 11/44.

The speed of screening is bound by the disk access, because of the nearly trivial computation and the amount of information to be transferred⁴⁴⁾, whereas the matching process is computation bound⁴⁵⁾.

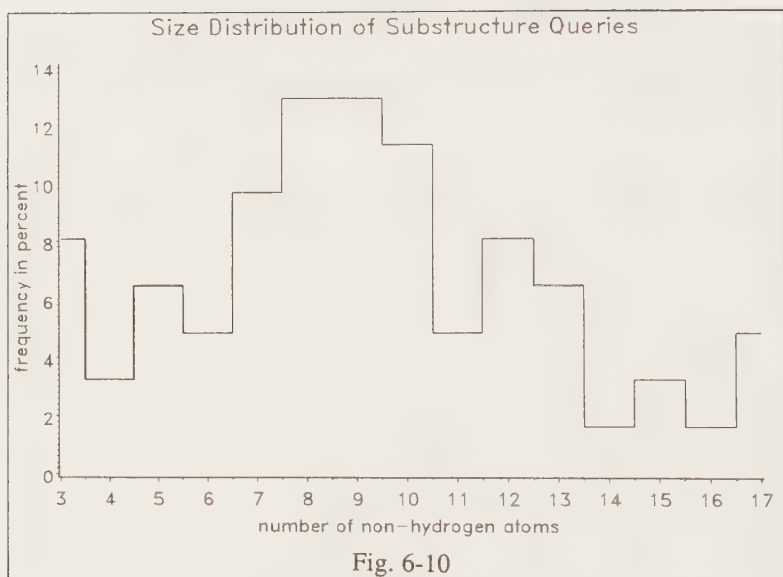
In the following, these figures will be related to the performance of GM-Search. Fig. 6-10 shows the distribution of non-hydrogen atoms in the substructure queries that were searched for in 1986 in the literature search service of the organic chemistry department at the University of Zürich. The average size of a query is 9.1 non hydrogen atoms.

The speed of GM-Search will be illustrated by a number of substructure searches in the sample structure library. An average query would not give reliable results, because too few candidates and matches are to be expected. Therefore, regularities in the set of structures were exploited. Fig. 6-11 shows the queries used in this investigation⁴⁶⁾. The results of the searches are summarized in Table VI-2.

44) Within 300 seconds 750,000 screen bit strings with 2,128 bits have to be read resulting in a transfer rate of 650 KBytes/second.

45) The data transfer rate is approx. 2.8 KBytes/second in the worst case.

46) Hetero atoms are depicted by the atom label Q and generic bonds are drawn as dotted lines.



The queries have different sizes as indicated by the number of non-hydrogen atoms. Using the measured computing times, the number of candidates and matches manageable within 5 minutes have been estimated for the queries of each type. The preprocessing time is not considered, because a host mainframe could be charged with it. The screening time seems to be dominated by disk seek time and latency because the amount of information to be read for each structure is less than 1/20 (96 bits/2,128 bits) of that used at CAS. The values that have been scaled up are the time spent to read the structure (although it is also affected by scattering of disk blocks) and the time for the substructure tests.

In CAS-Online, the worst case allowed has 25 % actual matches among the structures that pass the screening. The last two rows of Table VI-2 show the estimated limits to the number of candidates and matches, if this ratio is assumed and the query has a complexity similar to the one in the corresponding column. The numbers in parenthesis apply, if the GM-descrip-

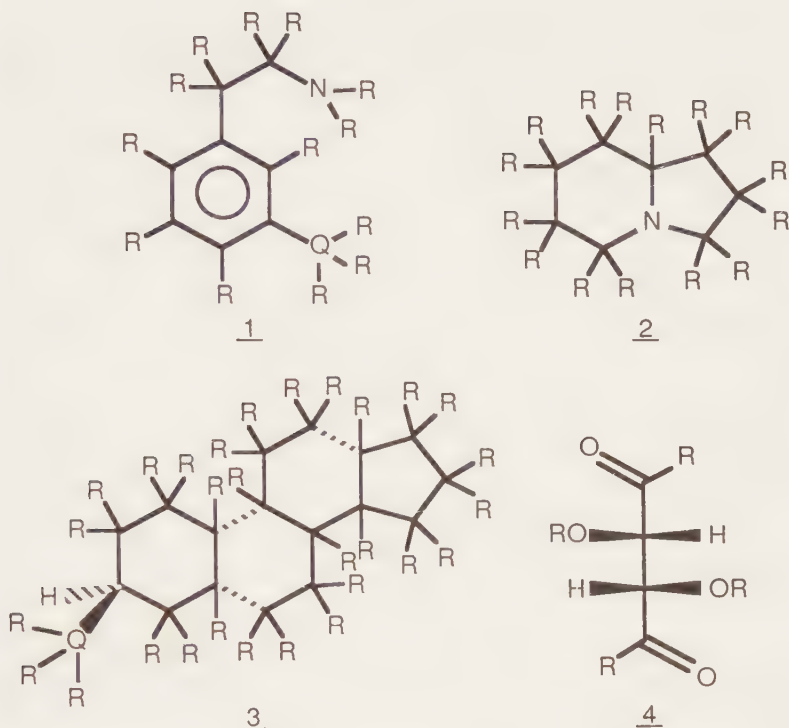


Fig. 6-11

tions of the structures are stored on disk in such a way, that read time can be neglected (consecutive blocks, only short seeks, and no directory accesses).

Since the number of candidates and matches are rather small the sample structure library has been used to estimate the average times of of successful and non-successful substructure tests. The average size of a structure in the sample structure library is 16.6 non-hydrogen atoms while an average CAS-structure contains 21.2 non-hydrogen atoms. Assuming that the computing time grows linearly with the structure size, which means that the time is mainly determined by the computation of the initial

Table VI-2

query	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>
non-H atoms	10	9	18	8
preprocessing	9.45 s	8.84 s	19.94 s	8.74 s
screening	1.82 s	1.81 s	1.81 s	1.81 s
reading	22.21 s	12.23 s	5.92 s	5.40 s
per structure	0.20 s	0.21 s	0.21 s	0.21 s
candidates	109	59	28	26
matches	72	27	11	6
matching successful	34.91 s	6.08 s	7.87 s	1.30
per structure	0.48 s	0.23 s	0.72 s	0.22 s
matching unsuccessful	5.66 s	3.07 s	8.76 s	2.08 s
per structure	0.15 s	0.10 s	0.52 s	0.10 s
estimated candidate	694	876	385	882
limit	(1290)	(2264)	(526)	(2308)
estimated match	143	219	96	221
limit	(323)	(566)	(132)	(577)

approximations of the center and unit mappings, or that the number of relaxation iterations is not affected by the structure size, the figures given in Table VI-2 have to be reduced by 22 %.

This analysis shows that the C-implementation of GM-Search (which is capable of handling stereochemistry) on an IBM-PC/AT-2 is about two to three times slower, in the worst case performance, than a pair of PDP 11 minicomputers at CAS doing substructure search only on the constitutional level.

7. Areas of Further Development

7.1. Improvements of GM-Search

In its present state, GM-Search stores only the first entry for each compound. This restriction was applied to avoid the programming associated with standard data base function and to concentrate on the algorithms dealing with GM-descriptions. Of course, if GM-Search should be developed into a production system, these facilities would have to be provided, for instance, by linking it with some standard data base subroutine package accessible from the C-language. Another improvement would be a non-graphical or a device-independent graphical interface for structure and query input, which would facilitate the implementation of the complete GM-Search on computers different from the PERQ-2.

Another topic is related to the structure and query input: The presently used graphical input system is only capable of dealing with tetrahedral stereocenters and double-bond stereoaxes. Therefore, the structures in the library of GM-Search are restricted to contain only those stereoelements. An obvious extension (which could be treated by GM-Search without any changes in its algorithms) would be the addition of structures containing other types of stereochemical relationships. Some of the conceivable stereoelements (or centrons [2]) could also be handled by a graphical input program, but then the potential users of the structure library would have to agree upon the amount of information to be stored for each stereoelement. This decision may be rather difficult. For example, the angle sizes to be selected for a description of the ferrocene molecule with its mutually rotating cyclopentadienyl ligands are not immediately evident. Floersheim [2] proposed a set of rules to govern this selection, but he was only concerned with the description of full structures, not substructures.

The performance of GM-Search is also susceptible to improvements. Especially the screen set, the screen generation algorithm, and the screen managers could be revised to optimize the screen-out and the speed of screen generation. The main sources of deficiency in screening are:

- Absence of non-matching augmented atom screens.
- Inadequacy of the Feldman and Hodes approximation of screen dependence if screens contain atom type descriptors of varying specificity.
- Small number of matching screens.

An extensive analysis of the screen set with respect to these three items should lead to improved screen-out without increasing the number of bits in the superimposed screen bit strings.

7.2. Scope of the Data Structure and the Key Algorithms

In addition to pure structure retrieval, the key algorithms in GM-Search, i.e. canonization and substructure testing, can also be used in other non-numerical applications of computer chemistry.

Similar to structure retrieval is reaction retrieval, which is of increasing interest to synthetic chemists. It can be formulated as a simultaneous search for two substructures being connected by the information, which atom of the starting material is to become which atom of the product. Even more adequately, this problem can be tackled by describing the whole reaction as a GM-description. Special units containing binary tuples represent the centers, that are related in starting material and product. The units belong to a new 'REACT' block and the permutation group for the tuples contains only the identity.

Prediction of spectral data also depends on a means to deal with stereochemical features. Especially NMR chemical shifts and coupling constants are influenced by relative positions of the atoms and mutual directions of the bonds involved. For the sake of simplicity of algorithms, the stereochemical environment is conventionally described in a hierarchical way [63]. This might be inadequate for some cases because, for instance, coupling constants depend on the mutual dihedral angles of the bonds. In rigid structures, they are determined largely by the ring system, which cannot be described in a locally hierarchical way. Therefore, a general substructure testing algorithm is necessary.

Computer aided synthesis planning is another topic where canonization and substructure testing algorithms are utilized. Canonization is used to find identities between two generated intermediates or between an intermediate and an available compound. Substructure testing is necessary to retrieve the retrosynthetically applicable reaction schemes. For synthesis planning, in contrast to structure retrieval, the substructure testing algorithm has to find all instances of a substructure within a structure. By setting a switch, the algorithm used in GM-Search can be adapted to achieve this. Symmetry equivalent matches lead to the generation of identical precursors, so that they should be avoided beforehand. Additional work would be necessary to let the matching algorithm of GM-Search benefit in this respect from the symmetry information collected during canonization.

The IUPAC rules for naming chemical compounds can, at least partly, be expressed in terms of substructure/naming-rule pairs. The framework of GM-Search could serve as means to store these rules in a form accessible to an expert system generating IUPAC-names for chemical structures. If the chemical structures could be input graphically on an inexpensive personal computer, this type of expert system would be of great utility to many chemists.

Structure-activity correlation is another field in which stereochemical substructure testing would be useful. While most current systems use either a geometrical (molecular modelling, docking, etc.) or graph theoretical (pattern recognition with graph indices) approach, stereochemically described substructures could serve as a means to bridge those two extremes. Compounds with a similar biological activity could be classified using common substructures as 'pharmacophores', which ought to contain constitutional, configurational, and, on a further level, conformational information. The first two aspects could be represented directly by GM-descriptions; for the representation conformations additional research is required.

7.3. Extensions to the Data Structure

One unsolved problem in structure retrieval is substructure search in (inorganic) crystal structures. In organic chemistry a similar problem arises with the description of polymers, where some 'unit cell' (the monomer) is repeated regularly many times. If only the repeating unit (the former monomer) is stored, substructure queries extending across monomer boundaries fail to give a match. It should be possible to extend the concept of GM-descriptions, and the algorithms manipulating them, to cope with these cases by using, for instance, some kind of 'modulo' arithmetic on the center numbers.

Related to the problem of describing repeating units is the representation of generic structures, which are common in the chemical patent literature. This problem has recently been studied by Lynch et al. [64-71], but they neglected stereochemical considerations. A solution of this problem in terms of some extended GM-descriptions would also provide a more compact coding of mixtures of isomers, especially racemic mixtures⁴⁷).

8. Summary of Major Results

In this work a data structure, called GM-description, has been developed which is capable of describing chemical structures as well as substructures in a homogeneous way. GM-descriptions are divided into units, each representing a small amount of structural information. Substructures may also contain certain generic attributes, such as sets of allowed bond orders for some bonds, or sets of possible atom types for some atoms.

Based upon this data structure, algorithms have been devised, which

- a) compute a unique code for a GM-description⁴⁸⁾ (canonization), and
- b) test if, for two GM-descriptions, one of them is contained in the other (substructure testing).

During canonization, generators for the automorphism group of the GM-description are also found and utilized to speed up the canonization process. The generators provide a concise description of the symmetry of the molecule represented by the GM-description.

Using these algorithms and a set of supportive procedures (for pruning, ring perception, and screening) a structure retrieval system, called GM-Search, was built that allows, from a given set of chemical structures, the retrieval of all those containing some query structure.

⁴⁷⁾ Presently, racemic mixtures must be represented as GM-descriptions of a compound structure comprising both enantiomers.

⁴⁸⁾ And thereby, it computes a unique name of the molecule or substructure represented by this GM-description.

An extrapolation of the computing times shows the feasibility of stereochemical substructure search using GM-Search even in a structure data base as large as the one maintained by Chemical Abstracts (over 7 million structures).

Appendix I: Glossary

Atom-by-Atom Matching:

The final operation in substructure testing to establish an atom-to-atom correspondence, retaining the relations of the substructure, between all atoms of the substructure and some of the structure to be tested.

Automorphism Group:

The group of isomorphisms of an algebraic structure on itself is called its automorphism group. In the case of a relational system, this group can be expressed as a permutation group acting upon the domain of the relational system.

Canonization:

The process of selecting a unique description from the possible ones is called the canonization of the description. Relational systems are canonized by uniquely numbering the elements of its domain up to the automorphisms.

Chemical Graph:

An undirected vertex- and edge-colored graph derived from a molecule by mapping the chemical bonds onto the graph edges, the atoms onto the graph vertices, and the bond orders and atom types onto the edge- and vertex-colors, respectively.

Complexity:

In the context of this dissertation, complexity of an algorithm means the dependence of the running time on the size of the input.

Generator:

The generators or the generating set of a group is a set of group elements which, by repeated multiplication with each other, produce every group element. Thus, every group element can be expressed as a string of these generators.

Generic Structure:

A generic structure is a generalization of the term substructure and is often written as a Markusch formula. It consists of a skeleton (or invariant part) and a set of substituents, which may or may not be present in a structure to be matched against it. It is possible to specify a set of allowed atom types for some atoms. This is also possible with GM-descriptions. If the substituents are not restricted to certain classes, and if there are no generic parts, the generic structure is equivalent to a substructure.

Hash Coding:

Given a set of keys k (in the structure retrieval context this is a canonical description of a molecule), a hashing function h computes a hash code $h_i = h(k_i)$ for each key k_i . If the hash codes of two keys are different, then the two keys must also be different. The set of possible keys (the set of possible molecules) is usually much larger than the set of possible hash codes and the number of actual keys. Thus, there are different keys with equal hash code.

Hash codes are used to assign storage locations to information units or records, or, as in this system, as a means to reduce the effort in examining a library of compounds, so that only the few

keys (canonical descriptions) with identical hash code have to be compared in detail.

NP-Complete:

For the problems belonging to the class of NP-complete problems, no algorithm with polynomial time complexity has been found. Therefore, those problems are generally considered to be computationally difficult. An example is the Hamiltonian path problem, which is the problem of finding a circuit in a graph, such that every node is visited exactly once.

Orbit:

The orbit of a member x of a set A , upon which a permutation group G is defined, is the set of all members of A that are mapped upon x by applying elements of G .

Parity Symbols:

Each stereoelement or stereogenic unit [72] of carbon stereochemistry only causes a doubling of the number of possible stereoisomers. For that reason, the usual description of stereoisomerism uses two-valued symbols (R/S, +1/-1, D/L, cis/trans, E/Z....), which are called parity symbols.

Query:

In database systems, the term query means the question posed to the system. In this work, query usually means a substructure. The structure retrieval system can be asked, if that substructure (query) is contained in any of the library structures.

Registry Number:

The registry numbers (RN) are used by Chemical Abstracts as unique identifiers for the more than 7 million registered chemical compounds. RNs are not computed, but assigned in the order of

registration. Many chemical companies also use these registry numbers to identify their chemicals.

Relational System:

In our research group, the relational system was developed to describe chemical structures completely, i.e. including potentially all stereochemical aspects. It is an extension of the graph concept, which has already been used to describe the constitution of molecules.

Screening:

Testing whether one structure is part of another structure is a time-consuming process. Therefore, nearly all substructure search systems use screens to minimize the effort. Screens are (usually small) substructures, which are tested for in every structure of the database. If one of the screens which is part of a the substructure in question, is not part of a structure, it can be inferred that the substructure is not contained in that structure. Since the screens are fixed substructures, this comparison can be done by mapping them onto a bit pattern and comparing only those.

Superimposed Coding:

Superimposed coding can serve to reduce the storage space required for the bit strings used in screening. These bit strings would otherwise contain a lot of zeros for all the screen substructures not present.

Appendix II: Sample Search

This appendix shows how a substructure search is performed with the PERQ-2 implementation of GM-Search (see also chapter 2). GM-Search is embedded into the FRAME program which also provides some other services like computation of CIP-descriptors or building a three dimensional model of a molecule.

After the FRAME program is entered, the screen contains the main command menu, shown on top of Fig. A-1, with a blank drawing area below it. The program is mainly controlled by a graphical input device called puck, which can be moved on a magnetostrictive tablet. The position of the puck is reflected by a graphics cursor, which changes shape as it is moved between drawing and command area. A command is input by moving the cursor to the command string and pressing one of the four buttons of the puck.

A substructure query is input by the subprogram DRAWEDITOR, which has the command menu shown in Fig. A-2. The user enters the DRAWEDITOR in the DRAW mode, which allows him to 'draw' the connections of the atoms of the query⁴⁹). In the drawing area the four buttons of the puck have different functions as depicted on the bottom right of Fig. A-2. One button is to start and continue a path of bonds, one button ends the path. Two other buttons are used to drag atoms or entire struc-

⁴⁹) A procedure similar to the one described here is used for the input of molecular structures to be added to the structure library.

DRAWEDITOR	CANDOR 1	REL SYS < RENH	GEOMETRIC DATA	HELP
CONGENERATOR	SEARCH SUB	RENH < REL SYS	HEXCEL	
CLEAN	UPDATE LB		MODEL BUILDER	DELETE STRUCTURE
GET	TO LIBRARY	REVERSE CP	CP	
PUT	FROM LIBRARY		MONETRAP	EXIT

Enter structure and query editor

Search the structure library

Write a text description of a structure

Version: 25.2.86
 Stereosymbols: Papermode
 Some numbers on
 Implicit H-atoms

Fig. A-1

[DRAW]	CHANGE SIZE	QUADRATIC	ALL ON	HELP
RESET COMP	ADJUST SIZE	HEXAGONAL	SOME ON	PLUT
		OFF	ALL OFF	
ADD COMPONENT	ROTATE x-Axis	SCROLL	Hs WIRE	GET DRAWING
ATOM TYPES	ROTATE y-Axis	EXPLOIT	DATE	PUT DRAWING
NUMBERS	ADJUST UNIT	STEREODRAW	PERMODE	DEL COPY/COMP
SPECIAL SYM	PUSH-PULL	CHARGE & RADICAL	RETURN	

Main draw command

Specify atom types

Enter stereo symbols and generic and aromatic bonds

Generic hetero atom

Optional free valence

Stereo Symbols

Meaning of the puck buttons
 in DRAW mode

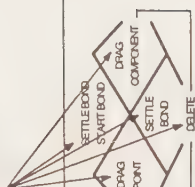


Fig. A-2

tures, respectively. The same two buttons can be used to delete a bond or an atom if the graphics cursor is positioned on the object and they are pressed simultaneously.

All atoms without a label are carbon atoms with the appropriate number of implicit hydrogen atoms attached to them. The atom type of an atom can be changed in the ATOM TYPES mode, to which the user can move by giving the ATOM TYPES command. The user can input the chemical symbol of an atom type and, by pointing at an atom and pressing a button, the atom changes to that type. In this mode it is also possible to specify explicit hydrogen atoms. The drawing of the query shows two special atom type symbols. 'Q' means that in the structures being searched this atom can be any hetero atom. 'R' designates an optional free valence of the query, i.e. the rest of the structure may be connected to the query part at this place.

The configuration of the tetrahedral centers is specified by SPECIAL SYM. which also allows input of aromatic and generic bonds. Each of the bond types is associated with a button or button combination and a bond is given such a bond type by moving the cursor to the bond and pressing the appropriate button. For stereosymbol bonds the small end of the wedge points to the atom closer to the cursor when the stereo bond is input.

After the query input, the DRAWEDITOR is left by the RETURN command and the main menu of Fig. A-1 is redisplayed. The substructure search can be started by issuing the SEARCH LIB. command. The output of the search for the query of Fig. A-1 is listed in Fig. A-3.

The first four lines are messages from the system about the operations being performed. The next block of lines shows the screens generated for the query followed by the resulting screen bit string in hexadecimal code. Each hexadecimal digit represents four bits of the bit string. After the screens have been computed, the Matcher tests each candidate that passed the screening if it contains the query. The answers are listed as pairs:

(<library number>) '<text string stored with the structure>'

The list of matches is followed by a message concerning computing time, number of matches(candidates), and the screening precision.

After the search the FRAME program waits for a carriage return and then redisplayes the query for further editing.

Loading Library Searcher

Pruning Hydrogens

Searching Rings

Generating Screens

```
Adding screen: EC 1 N H0 H1 H2 H3 H4 H5
Adding screen: AB 1 C-C-C-C
Adding screen: AB 1 N-C-C-C
Adding screen: AB 1 N?C
Adding screen: AB 1 Q-C-C-C
Adding screen: AB 1 Q-C-C-Q
Adding screen: AB 1 Q?C?C?Q
Adding screen: 'C-CHQ-C group'
Adding screen: 'C-CHQ-C?Q'
Adding screen: 'C-CHQ-CHQ-C'
Adding screen: '(threo)-C-(S)CHQ-(S)CHQ-C'
```

screen bit string = 101800001000110901500000

Matching

```
(207) 'bs.12.2'
(215) 'bs.13.1'
(218) 'bs.13.12'
(221) 'bs.13.15'
(225) 'bs.13.5'
(663) 'hesse.5.122 (+)-Rhoeadin'
(679) 'hesse.5.138 (-)-Ophiocarpin'
(730) 'hesse.5.192 Febrifugin'
(731) 'hesse.5.196 (-)-Retroncanol'
(734) 'hesse.5.199 Monocrotalin'
(792) 'hesse.5.220 (-)-Ephedrin'
(792) 'hesse.5.256 Carolinianin'
(837) 'hesse.5.65' (+)-Dehydrovoachalotin'
```

Total setup time = 15599 ms, search time= 2100 ms, read time = 8653 ms,
match time = 4296 ms, and non-match time = 5723 ms for 13(47) matches.
Total time = 36371 ms, screening precision = 27.66 %.
Enter <cr> to proceed

Fig A-3

References

- [1] D. H. Rouvray (1976), The Topological Matrix in Quantum Chemistry, in "Chemical Application of Graph Theory", Academic Press, London New York San Francisco.
- [2] P. Floersheim, Dissertation University of Zürich, 1986. From Descriptions of Molecules to their Symmetry.
- [3] P. Floersheim, K. Wirth, M. K. Huber, D. Pazis, F. Sigrist, H. R. Haegi, A. S. Dreiding, Studies in Physical and Theoretical Chemistry, 23, 59 (1983). From Mobile Molecules to their Symmetry Groups: A Computer-Implemented Method.
- [4] K. Wirth, M. K. Huber, MATCH 12, 3 (1981). Numbering of Finite Relational Systems
- [5] P. G. Dittmar, N. A. Farmer, W. Eisanick, R. C. Haines, J. Mockus, J. Chem. Inf. Comput. Sci. 23, 93 (1983). The CAS ONLINE Search System. 1. General System Design and Selection, Generation, and Use of Search Screens.
- [6] J. Mockus, A. C. Isenberg, G. G. van der Stouw, J. Chem. Inf. Comput. Sci. 21, 183 (1981). Algorithmic Generation of Chemical Abstracts Index Names. 1. General Design
- [7] CAS ONLINE Screen Dictionary for Substructure Search, 2nd Edition, Columbus Ohio (1981).
- [8] J. P. Moosemiller, A. W. Ryan, R. E. Stobaugh, J. Chem. Inf. Comput. Sci. 20, 83 (1980). The Chemical Abstracts Service Chemical Registry System. VIII. Manual Registration
- [9] J. E. Blackwood, P. M. Elliott, R. E. Stobaugh, C. E. Watson, J. Chem. Inf. Comput. Sci. 17, 3 (1977). The Chemical Abstracts Service Chemical Registry System. III. Stereochemistry

- [10] H. L. Morgan, J. Chem. Doc. 5, 107 (1965). The Generation of a Unique Machine Description for Chemical Structures -A Technique Developed at Chemical Abstracts Service.
- [11] J.-E. Dubois, Y. Sobel, J. Chem. Inf. Comput. Sci., 25, 326 (1985). DARC System for Documentation and Artificial Intelligence in Chemistry
- [12] R. Attias, J. Chem. Inf. Comput. Sci. 23, 102 (1983). DARC Substructure Search System: A New Approach to Chemical Information
- [13] J. E. Dubois, A. Panaye, P. Cayzergues, C. R. Seances Acad. Sci., Ser. 2, 295, 135 (1982). DARC code. Assigning planar Stereoisomers by a Stereo Substructure (S₃) with Seven Points.
- [14] P. Cayzergues, A. Panaye, J.-E. Dubois, C. R. Seances Acad. Sci., Ser. C, 290, 441 (1980). DARC code. Description of Complex Stereocenters.
- [15] J.-E. Dubois, A. Panaye, P. Cayzergues, C. R. Seances Acad. Sci., Ser. C, 290, 429 (1980). Code DARC. Choix des Orions en stéréochimie dans l'espace 3D.
- [16] E. Zass, S. Müller, Chimia, 40, 38 (1986). Neue Möglichkeiten zur Recherche von organisch-chemischen Reaktionen: Ein Vergleich der 'in-house'-Datenbanksysteme REACCS, SYNLIB und ORAC.
- [17] G. W. Adamson, J. M. Bird, G. Palmer, W. A. Warr, J. Chem. Inf. Comput. Sci., 25, 90 (1985). Use of MACCS within ICI.
- [18] S. Anderson, J. Mol. Graphics, 2, 83 (1984). Graphical Representation of Molecules and Substructure-Search Queries in MACCS.
- [19] R. Custer, Dissertation, Swiss Federal Institute of Technology, Zürich, 1985. An Investigation of the CIP-System, a Mobile Molecular Model, and Computer Implementations.
- [20] "Contents of Beilstein, Representative Examples", Springer Verlag, Berlin, Heidelberg New York, 1983.
- [21] M. Hesse, "Alkaloidchemie", Georg Thieme Verlag, Stuttgart, 1978.
- [22] "Optically Active Compounds", JPS Chimie, CH-2022 Bevaix, Catalogue 1986.

- [23] J. Mockus, R. E. Stobaugh, J. Chem. Inf. Comput. Sci. 20, 18 (1980). The Chemical Abstracts Service Chemical Registry System. VII. Tautomerism and Alternating Bonds.
- [24] A. Feldman, J. Chem. Inf. Comput. Sci. 17, 220 (1977). An Efficient Design for Chemical Structure Searching. III. The Coding of Resonating an Tautomeric Forms
- [25] Moore, E. F., The Shortest Path Through a Maze, Proc. Internat. Symp. Switching Th., 1957, Part II, p. 285, Harvard Univ. Press, 1959.
- [26] M. Bersohn, J. Comput. Chem., 4, 110(1983). A Fast Algorithm for the Calculation of the Distance Matrix of a Molecule.
- [27] R. C. Read, D. G. Corneil, J. Graph Theor., 1, 339 (1977). The Graph Isomorphism Disease.
- [28] E. Luks, Proc. 21st IEEE Symp. Found. Comp. Sci., Rochester, 42-49 (1980). Isomorphism of Graphs of Bounded Valence can be Tested in Polynomial Time.
- [29] K. Wirth, J. Chem. Inf. Comput. Sci., 26, 242 (1986). Coding of Relational Descriptions of Molecular Structures.
- [30] J. B. Hendrickson, A. G. Toczko, J. Chem. Inf. Comput. Sci., 23, 171 (1983). Unique Numbering and Cataloguing of Molecular Structures.
- [31] M. Uchino, J. Chem. Inf. Comput. Sci., 22, 201 (1982). Algorithms for the Unique and Unambiguous Coding and Symmetry Preception of Molecular Structure Diagrams. 5. Unique Coding by the Mothod of 'Orbit Graphs'.
- [32] M. Randic, G. M. Brissey, C. L. Wilkins, J. Chem. Inf. Comput. Sci., 21, 52 (1981). Computer Perception of Topological Symmetry via Canonical Numbering of Atoms.
- [33] M. Uchino, J. Chem. Inf. Comput. Sci., 20, 124 (1980). III. Method of Subregion Analysis for Unique Coding and Symmetry Perception.
- [34] M. Uchino, J. Chem. Inf. Comput. Sci., 20, 121 (1980). II. Basic Algorithm for Unique Coding and Computation of Symmetry Group.

- [35] M. Uchino, J. Chem. Inf. Comput. Sci., 20, 116 (1980). I. Vector Functions for Automorphism Partitioning.
- [36] C. A. Shelly, M. E. Munk, J. Chem. Inf. Comput. Sci., 19, 247 (1979). An Approach to the Assignment of Canonical Connection Tables and Topological Symmetry Perception.
- [37] M. Randic, J. Chem. Inf. Comput. Sci., 17, 171 (1977). On Canonical Numbering of Atoms in a Molecule and Graph Isomorphism.
- [38] K. K. Agarwal, Ph. D. Thesis, State University of New York at Stony Brook (1976). Transformation and Canonization Algorithms for Grapg Representable Structures with Applications to a Heuristic Program for the Synthesis of Organic Molecules.
- [39] C. Jochum, J. Gasteiger, J. Chem. Inf. Comput. Sci., 17, 113 (1977). Canonical Numbering an Constitutional Symmetry.
- [40] W. T. Wipke, T. M. Dyott, J. Am Chem. Soc., 96, 4834 (1974). Stereochemically Unique Naming Algorithm.
- [41] F. Choplin, R. Marc, G. Kaufmann, W. T. Wipke, J. Chem Inf. Comput. Sci, 18, 110 (1978). Computer Design of Synthesis in Phosphorous Chemistry: Automatic Treatment of Stereochemistry.
- [42] C. J. Colburn, Proc. 10th SE Conf. Comb., Graph Theor., Comput., 281 (1979). Refinement Techniques for Graph Isomorphism.
- [43] B. D. McKay, Lecture Notes in Math., 686, 223 (1978). Computing Automorphisms and Canonical Labellings of Graphs.
- [44] C. M. Hoffmann, "Group-Theoretic Algorithms and Graph Isomorphism", Lecture Notes in Computer Science, 1982, Springer Verlag, Berlin-Heidelberg-New York.
- [45] C. M. Hoffmann, "Group-Theoretic Algorithms and Graph Isomorphism", p. 114, Lecture Notes in Computer Science, 1982, Springer Verlag, Berlin-Heidelberg-New York.
- [46] A. V. Aho, J. E. Hopcroft, J. D. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley (1974).
- [47] A. Lingas, Lecture Notes in Comput. Sci., 112, 290 (1981). Certain Algorithms for Subgraph Isomorphism Problems.

- [48] A. K. Mackworth, Artificial Intelligence, 8, 99 (1977). Consistency in Networks of Relations.
- [49] R. M. Haralick, L. G. Shapiro, IEEE Trans. Pat. Anal. Mach. Intell., Vol. PAMI-1, 173(1979). The Consistent Labeling Problem: Part I.
- [50] L. C. Ray, R. A. Kirsch, Science, 126, 814(1957). Finding Chemical Records by Digital Computer.
- [51] E. H. Sussenguth, J. Chem. Doc., 5, 36(1965). A Graph-Theoretical Algorithm for Matching Chemical Structures.
- [52] J. Figueras, J. Chem. Doc., 12, 237(1972). Substructure Search by Set Reduction.
- [53] J. K. Cheng, T. S. Huang, Pattern Recognition, 13, 371(1981). A Subgraph Isomorphism Algorithm Using Resolution.
- [54] L. Kitchen, A. Rosenfeld, IEEE Trans. Syst, Man, Cybern., vol. SMC-9, 869(1979). Discrete Relaxation for Matching Relational Structures.
- [55] T.-K. Ming, S. J. Tauber, J. Chem. Doc. 11, 47 (1971). Chemical Structure and Substructure Search by Set Reduction.
- [56] W. Graf, H. K. Kandl, H. Kniess, B. Schmidt, R. Warszawski, J. Chem. Inf. Comput. Sci. 19, 51 (1979). Substructure Retrieval by means of the BASIC Fragment Search Dictionary Based on the Chemical Abstracts Service Chemical Registry System III
- [57] R. E. Stobaugh, J. Chem. Inf. Comput. Sci. 20, 76 (1980). The Chemical Abstracts Service Chemical Registry System. VI. Substance Related Statistics
- [58] C. N. Mooers, Am. Doc.. 2, 20(1951). Zatocoding Applied to Mechanical Organisation of Knowledge.
- [59] D. E. Knuth, "The Art of Computer Programming. Vol. III. Sorting and Searching", Addison-Wesley, Reading Mass. 1973, p. 559.
- [60] A. Feldman, and L. Hodes, J. Chem Inf. Comput. Sci., 15, 147(1975). An Efficient Design for Chemical Structure Searching. I. The Screens.

- [61] L. Hodes, J. Chem. Inf. Comput. Sci., 16, 88(1976). Selection of Descriptors According to Discrimination and Redundancy. Application to Chemical Substructure Searching.
- [62] L. Hodes, A. Feldman, J. Chem. Inf. Comput. Sci., 18, 96 (1978). An Efficient Design for Chemical Structure Searching. II. The File Organization.
- [63] C. W. Crandell, N. A. B. Gray, D. H. Smith, J. Chem. Inf. Comput. Sci., 22, 48 (1982). Structure Evaluation Using Predicted ^{13}C Spectra.
- [64] M. F. Lynch, J. M. Barnard, S. M. Welford, J. Chem. Inf. Comput. Sci. 21, 148-150 (1981). Computer Storage of Generic Chemical Structures in Patents. 1. Introduction and General Strategy.
- [65] J. M. Barnard, M. F. Lynch, S. M. Welford, J. Chem. Inf. Comput. Sci. 21, 151-161 (1981). Computer Storage of Generic Chemical Structures in Patents. 2. GENSAL, a Formal Language for the Description of Generic Chemical Structures.
- [66] S. M. Welford, M. F. Lynch, J. M. Barnard, J. Chem. Inf. Comput. Sci. 21, 161-168 (1981). Computer Storage of Generic Chemical Structures in Patents. 3. Chemical Grammars and Their Role in the Manipulation of Chemical Structures.
- [67] J. M. Barnard, M. F. Lynch, S. M. Welford, J. Chem. Inf. Comput. Sci. 22, 160-164 (1982). Computer Storage of Generic Chemical Structures in Patents. 4. An Extended Connection Table Representation for Generic Structures.
- [68] S. M. Welford, M. F. Lynch, J. M. Barnard, J. Chem. Inf. Comput. Sci. 24, 57-66 (1984). Computer Storage of Generic Chemical Structures in Patents. 5. Algorithmic Generation of Fragment Descriptors for Generic Screening.
- [69] J. M. Barnard, M. F. Lynch, S. M. Welford, J. Chem. Inf. Comput. Sci. 24, 66-71 (1984). Computer Storage of Generic Chemical Structures in Patents. 6. An Interpreter Program for the Generic Structure Description Language GENSAL.

- [70] A. von Scholley, J. Chem. Inf. Comput. Sci. 24, 234-241 (1984). A Relaxation Algorithm for Generic Chemical Structure Screening.
- [71] V. J. Gillet, S. M. Welford, M. F. Lynch, P. Willett, J. M. Barnard, G. M. Downs, G. Manson, J. Thompson, J. Chem. Inf. Comput. Sci. 26, 118-126 (1986). Computer Storage of Generic Chemical Structures in Patents. 7. Parallel Simulation of a Relaxation Algorithm for Chemical Substructure Search.
- [72] R. S. Cahn, C. K. Ingold, V. Prelog, Angew. Chem., 78, 413 (1966). Spezifikation der molekularen Chiralität.

Curriculum

Name: Bernhard Rohde

Date: February 16, 1956

Place of birth: Mücke Hessen/BRD

1962-1966	Elementary School at Grünberg-Lehnheim/Hessen, BRD
1966-1973	Gymnasium at Grünberg/Hessen/BRD
1973-1982	Study of Chemistry and Mathematics at the Justus-Liebig University of Giessen, BRD
1982	Diploma thesis: "Simulation des ESR-Spektrums des Allyl-Radikals in einer polykristallinen Argonmatrix" Supervisor: Prof. G. Maier
1982-1987	Work on computer-oriented descriptions of molecules Supervisor: Prof. A. S. Dreiding
1984-1987	Assistant (in charge of computer-assisted retrieval of chemical information)
1987	Doctoral examination

Courses and seminars attended during my education were held by:

University of Zürich: Professors A. S. Dreiding, W. von Philipsborn, W. Reichhart, and Dr. M. Karpf. University of Giessen: Professors R. Askani, C. Fenske, S. Filippi, R. Gruehn, N. Grün, L. Hoischen, R. Hoppe, W. Luh, G. Maier, R. Michel, U. Mosel, U. Ott, A. Scharmann, W. Seidel, and M. Winnewisser.

